NOVA ABORDAGEM DE ALGORITMO MEMÉTICO APLICADO A PROBLEMAS DE OTIMIZAÇÃO NÃO-LINEAR

LEANDRO DOS SANTOS COELHO

Laboratório de Automação e Sistemas, Centro de Ciências Exatas e de Tecnologia
Pontificia Universidade Católica do Paraná
Rua Imaculada Conceição, 1155, CEP 80215-030, Curitiba, PR, Brasil
e-mail: lscoelho@rla01.pucpr.br

Resumo— Este artigo propõe duas abordagens de metodologias de otimização usando algoritmos meméticos para a resolução de problemas de otimização não-linear. As abordagens apresentadas tratam de problemas que necessitam de uma representação mista das soluções, ou seja, uma representação tanto de variáveis contínuas (parâmetros de ponto flutuante) quanto variáveis discretas (parâmetros inteiros). As abordagens propostas são: (i) algoritmo genético canônico (otimização de parâmetros inteiros) atuando em paralelo com programação evolutiva combinada com algoritmo *simulated annealing* (otimização de parâmetros de ponto flutuante), e (ii) algoritmo genético canônico (otimização de parâmetros inteiros) atuando em paralelo com programação evolutiva combinada com o método simplex de Nelder e Mead (otimização de parâmetros de ponto flutuante). O desempenho destes algoritmos meméticos para o projeto de um vaso de pressão é apresentado e discutido.

Abstract— This paper proposes two approaches of optimization methodologies using memetic algorithm for the resolution of nonlinear optimization problems. The presented approaches treat problems with mixed coding to representation of the continuous (floating-point parameters) and discrete (integer parameters) variables. The approaches are: canonical genetic algorithm (integer parameters optimization) and hybrid evolutionary programming with simulated annealing (floating-point parameters optimization), and (ii) canonical genetic algorithm (integer parameters optimization) and hybrid evolutionary programming with Nelder-Mead simplex method (floating-point parameters optimization). Performance of these memetic algorithms for the design of a pressure vessel is presented and discussed.

Keywords— evolutionary computation; memetic algorithms; optimization; evolutionary programming; simulated annealing.

1 Introdução

Os métodos de otimização apresentam usualmente duas formas de configuração: os métodos determinísticos e os métodos estocásticos. Os algoritmos evolutivos (*AEs*) — metodologias da área computação evolucionária ou evolutiva — não são algoritmos computacionais em seu significado usual, mas formam uma classe de métodos estocásticos regidos por princípios similares.

Nos *AEs*, um conjunto de soluções (população) é manipulado a cada iteração, em contraste com outros métodos de otimização, onde apenas uma solução para o problema é utilizada a cada iteração. A chance de que um indivíduo da população seja selecionado na próxima geração depende da função de aptidão (*fitness*) do indivíduo, que consiste, geralmente, de uma função objetivo ou mesmo uma transformação simples desta para o tratamento do problema em questão. Um compromisso entre convergência (*explotation*) e diversidade dos membros que constituem a população (*exploration*) é um problema constante em *AEs* e deve ser considerado na configuração de uma metodologia de otimização eficiente. Os *AEs* são especialmente úteis em tarefas de otimização global, onde os métodos determinísticos podem levar a soluções de mínimos locais. Entretanto, a configuração de abordagens compostas por técnicas híbridas (busca global e local) de otimização é uma alternativa relevante tratada na literatura (Yen *et al.*, 1998; Tsutsui *et al.*, 1999; Coelho & Coelho, 1999).

Para obter-se os benefícios de uma configuração híbrida, uma forma eficiente é executar, inicialmente, um *AE* para localizar a região de ótimo global e após aplicar-se outra metodologia de otimização para a busca local. Esta abordagem pode ser denominada de *algoritmo memético*. O termo *algoritmo memético* foi introduzido por Moscato & Norman (1992) para descrever um *AE* que apresenta um procedimento de busca local. Este procedimento de busca local é usualmente executado por outra técnica de otimização. O termo foi motivado pela noção de *memes* proposta por Dawkins (1976).

Este artigo propõe a implementação de duas abordagens de algoritmos meméticos. A eficiência destas abordagens de algoritmos meméticos é analisada em um problema de otimização não-linear da área de engenharia mecânica.

O artigo é organizado da seguinte forma. Na seção 2 são apresentadas noções, características e algoritmos para o projeto dos algoritmos meméticos. A descrição de problemas de otimização não-linear é apresentada é abordada na seção 3. As simulações e análise dos resultados obtidos da aplicação de algoritmos meméticos são apresentados na seção 4. A conclusão e as perspectivas de futuros trabalhos são abordadas na seção 5.

2 Algoritmos meméticos

A seguir são apresentadas as abordagens de algoritmos meméticos tratadas neste artigo. As abordagens são: (i) *AG* (otimização de parâmetros

inteiros) em conjunto com *PE* híbrida com algoritmo *simulated annealing* (otimização de parâmetros de ponto flutuante), e (ii) *AG* (otimização de parâmetros inteiros) em conjunto com *PE* híbrida com o método simplex de Nelder e Mead (otimização de parâmetros de ponto flutuante). O diagrama de fluxo da informação nestes algoritmos meméticos baseados em uma *PE* é apresentado na figura 1.

A vantagem da utilização de um método de busca direto na otimização de parâmetros de ponto flutuante, por exemplo, método simplex e/ou *simulated annealing* (Kirkpatrick *et al.*, 1983) em relação a busca local está em uma maior velocidade de convergência do método de busca como um todo quando combinado com uma *PE*.

2.1 Otimização de parâmetros inteiros

2.1.1 Algoritmos genéticos (AGs)

Um *AG* canônico (representação binária) caracteriza-se por: (i) operar em uma população de pontos, (ii) não requerer cálculos de derivadas e informação sobre o gradiente da função objetivo, (iii) trabalhar com a codificação de seu conjunto de parâmetros, não com os próprios parâmetros (representação binária), (iv) realizar transições probabilísticas, em vez de regras determinísticas, (v) necessitar apenas da informação sobre o valor da função objetivo para cada indivíduo da população, (vi) apresentar simplicidade conceitual, e (vii) ser pouco afetado, quanto à eficiência, quando descontinuidades e ruídos estão presentes nos dados do problema.

Em um *AG*, uma população de soluções-candidatas é aleatoriamente gerada e "evolui" para uma solução do problema, através da aplicação dos operadores genéticos de seleção, recombinação e mutação. Um *AG* pode ser projetado, na sua configuração básica, conforme as seguintes etapas:

(i) gerar a população inicial de parâmetros compreendendo *Nind* indivíduos (soluções para o problema). Cada uma das soluções consiste de vetores $x_i \in \{0,1\}$ (representação canônica). Estes parâmetros são iniciados aleatoriamente, de acordo com uma distribuição uniforme;
(ii) classificar cada solução x_i, i=1,...,*Nind*, com relação ao cálculo da função de aptidão, ou seja, avalia-se o grau de adaptação de cada indivíduo da população em relação ao problema;
(iii) selecionar os indivíduos mais aptos de acordo com uma estratégia de seleção;
(iv) aplicar o operador genético de cruzamento;
(v) aplicar o operador genético de mutação;
(vi) gerar uma nova população; e
(vii) repetir as etapas (ii) a (vi) até que um critério de parada seja satisfeito.

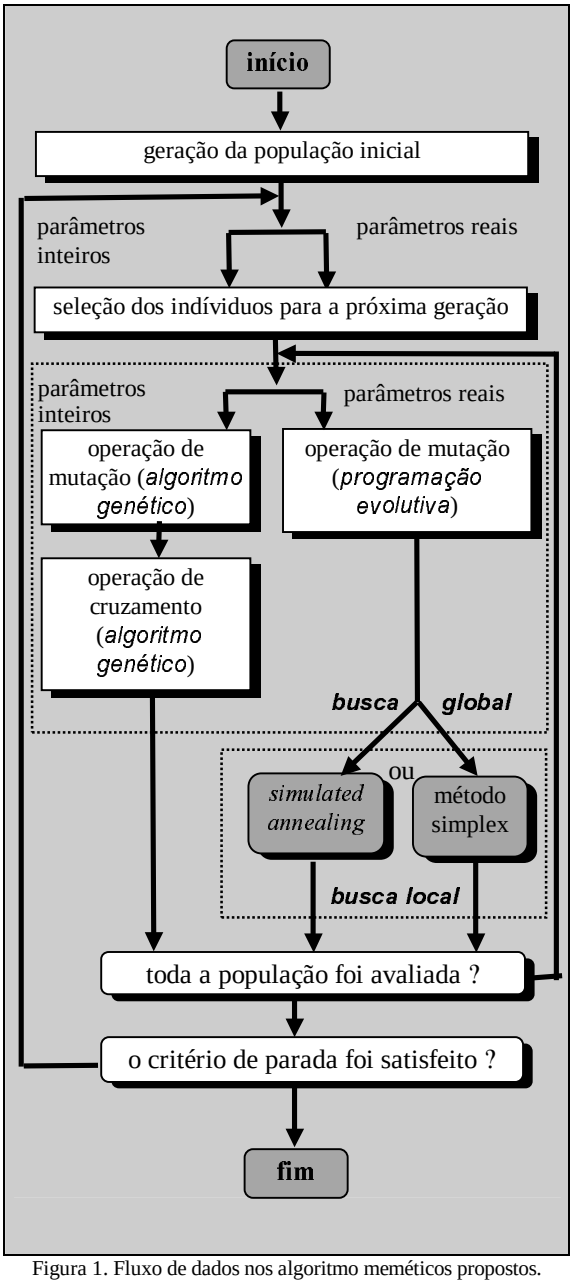

Figura 1. Fluxo de dados nos algoritmo meméticos propostos.

2.2 Otimização dos parâmetros reais

2.2.1 Programação evolutiva (PE)

Em uma *PE*, a população inicial de soluções (valores reais) é gerada aleatoriamente de acordo uma função densidade, sendo após toda população classificada em relação a um dado objetivo. As soluções-descendentes são geradas de soluções-ancestrais através de operações de mutação. Tipicamente, cada indivíduo é composto de uma variável-objeto (vetor solução) acompanhada de um desvio padrão. Neste artigo a *PE* tem o indivíduo modificado por uma variável aleatória com distribuição de Cauchy (Yao & Liu, 1996; Chellapilla, 1998). A utilização do operador de mutação com distribuição de Cauchy necessita de uma função densidade de probabilidade centrada na origem e definida por

$$f_t = \frac{1}{\pi}\frac{t}{t^2 + x^2}, \qquad para \ -\infty < x < \infty \tag{1}$$

onde $t > 0$, é um parâmetro de escala. A função de distribuição correspondente é:

$$F_t(x) = \frac{1}{2} + \frac{1}{\pi} arctan\left(\frac{x}{t}\right) \tag{2}$$

A forma de $f_t(x)$ parece-se com a função de densidade Gaussiana (normal), mas nas proximidades o eixo $f_t(x)$ decresce mais vagarosamente. Como um resultado disto, a variância da distribuição de Cauchy é infinita. O operador de mutação, com distribuição de Cauchy é útil para escapar de ótimos locais. A figura 2 mostra as funções densidade de Cauchy e Gaussiana com média zero e desvio padrão, σ.

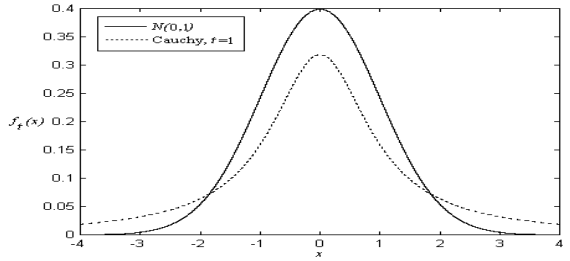

Figura 2. Funções densidade de Cauchy e Gaussiana.

O algoritmos memético proposto é constituído de uma *PE* híbrida com (i) algoritmo *simulated annealing* ou (ii) método simplex são implementados da seguinte forma:

(i) A população inicial de parâmetros a serem otimizados compreeende *nv* parâmetros. Cada uma das soluções consiste de vetores $x_i \in \Re^+$ e σx_i, onde $i=\{1,...,nv\}$, com suas dimensões correspondendo ao valor de *nv* do controlador. Os componentes de cada x_i, $i=\{1,...,nv\}$, são selecionados de acordo com uma distribuição uniforme no intervalo $[0;\kappa]$. Os componentes de σ_{xi} são selecionados de acordo com uma distribuição uniforme no intervalo $[0;\psi]$;

(ii) Cada solução, x_i, $i=\{1,...,nv\}$, é classificada com relação a função de adaptação *F(J)*;

(iii) Cada indivíduo ancestral (x_i, σ_{xi}'), $i=\{1,...,nv\}$, gera um descendente (x_i', σ_{xi}'), através da equação:

$$x_i'(j) = x_i(j) + \delta_j(0, \sigma'_{xi}(j)), \qquad j=\{1,...,nv\} \tag{3}$$

onde $x_i(j)$, $x_i'(j)$, $\sigma_{xi}(j)$, e $\sigma_{xi}'(j)$ denotam o j-ésimo componente dos vetores x_i, x_i', σ_{xi}, e σ_{xi}', respectivamente; $N(\mu,\sigma)$ denota uma variável aleatória Gaussiana com média μ e desvio padrão σ; $N_j(\mu,\sigma)$ indica que o número aleatório é gerado um novo para cada valor de j, δ_j é uma variável aleatória de Cauchy centrada em zero com parâmetro de escala de 1. Os fatores τ e τ' são usualmente configurados para $1/\left(\sqrt{2*nv}\right)$ e $1/\left(\sqrt{2\sqrt{nv}}\right)$, respectivamente;

(iv) Cada vetor descendente x_i', $i=\{1,...,nv\}$, é avaliado com relação a uma função de adaptação *F(J)*;

(v) As comparações são conduzidas sobre todas as x_i e x_i' soluções, $i=\{1,...,nv\}$. Para cada solução, 10 oponentes (seleção por torneio) são selecionados aleatoriamente dos vetores ancestrais (x_i) e descendentes (x_i') com igual probabilidade. Em cada comparação, se a solução condicionada oferece pelo menos um desempenho tão adequado quanto o oponente selecionado aleatoriamente, então recebe uma "vitória";

(vi) As soluções de x_i e x_i', $i=\{1,...,nv\}$, que possuem mais vitórias são selecionadas para serem pais na próxima população, sendo que os vetores a eles associados são também incluídos;

(vii) Aplicação do algoritmo *simulated annealing* ou simplex de Nelder e Mead (descritos a seguir); e

(viii) Enquanto um critério de parada não é satisfeito, o ciclo evolutivo retorna ao passo (iii). O critério de parada adotado neste artigo é de 200 gerações.

PE híbrida com *simulated annealing*: O *SA* é uma técnica de otimização estocástica de propósito geral que tem provado ser eficiente na aproximação de ótimos globais em diferentes problemas combinatoriais NP-*hard*. O algoritmo *simulated annealing* (*SA*) — ou têmpera simulada — é uma variação de algoritmos de subida de encosta, onde o objetivo é a minimização do nível de energia. Segundo Sun (1995), o *SA* difere dos algoritmos de otimização convencionais devido ao fato que ele pode: (i) processar funções-objetivo que possuem graus arbitrários de não-linearidades, descontinuidades e estocasticidade, (ii) tratar restrições impostas pela função-objetivo, (iii) ser implementado facilmente (poucas linhas de código) em relação a outros algoritmos de otimização não-linear, e (iv) garantir estatisticamente que encontra a solução ótima.

A abordagem do *SA* é probabilística, não requer informação de derivadas e não é afetada por descontinuidades e não-linearidades. O *SA* é um procedimento de descida de encosta, mas configurado de forma modificada. O *SA* realiza pequenos passos de busca em uma topografia local; se o passo resulta em uma melhora na solução, a nova solução é aceita, caso contrário, esta nova solução é aceita com probabilidade que inicialmente é configurada para 1. Contudo, com o progresso das iterações o *SA* é reduzida a probabilidade de aceitar uma nova solução que não apresenta aprimoramento em relação àquela obtida na iteração anterior.

A simulação a uma temperatura fixa, *T*, consiste em uma seqüência de passos. A cada passo, é dado um pequeno deslocamento a um dos átomos e é calculada a variação da energia, ΔE, que o sistema sofre com aquele deslocamento. Se $\Delta E \leq 0$, o deslocamento é incorporado ao estado do sistema,

que é utilizado para o passo seguinte, contudo se $\Delta E \geq 0$, a aceitação ou não da transição para um estado maior de energia tem uma probabilidade dada pela função

$$p = e^{-\Delta E * k_b * T} \quad (4)$$

onde k_b é a constante da distribuição de Boltzmann, T é a temperatura e ΔE é a mudança no nível de energia. Com a finalidade de simplificar a implementação do *SA* pode-se ignorar a constante k_b e reescrever a probabilidade de aceitação de uma configuração de energia mais alta como

$$p = e^{-\left(\frac{\Delta E}{T}\right)} \quad (5)$$

A velocidade com que o sistema é resfriado (cronograma de têmpera) é dada pela equação:

$$r(k) = \alpha_{SA}^{(k+1)} T_k \text{, para todo } k \geq 0 \quad (6)$$

O procedimento de *annealing* é aplicado a *PE*, a cada geração, após as operações de recombinação e mutação, para sintonização fina dos valores dos indivíduos do *AE*. O *SA* é aplicado, a cada geração, após as operações de recombinação e mutação, através de uma pequena perturbação no valor dos indivíduos. Um baixo fator de temperatura, T, e um rápido fator de *annealing*, α_{SA}, são utilizados no algoritmo memético composto de *PE* híbrida com *SA* (Tan *et al.*, 1995).

<u>*PE* híbrida com método simplex:</u> O método *simplex* de Nelder & Mead (1965) — também conhecido como método do politopo ou método ameba — é uma técnica de busca direta utilizada em problemas de otimização. Uma busca direta significa que o método é guiado somente pelo cálculo do valor da função em vários pontos e não necessita da avaliação da primeira e segunda derivadas (parcial) da função a ser minimizada (ou maximizada).

O método *simplex* "mantém" diversas soluções diferentes durante o procedimento de otimização. Esta é uma característica similar aos *AEs* que, no caso, mantêm uma população de indivíduos. A maior diferença entre os *AEs* e o método *simplex*, é que o método *simplex* escolhe os soluções de forma determinística, configurando um politopo de forma a "repelir" soluções inadequadas. Em contrapartida, os *AEs* têm desempenho vinculado à geração de novas soluções para o problema através dos indivíduos com melhores valores de função de aptidão.

O método simplex apresenta sua configuração básica baseada na definição de um *simplex*. Um simplex é uma estrutura formada por (*nptos*+1) pontos em um espaço multidimensional. O método adotado aqui apresenta algumas modificações do método original de Nelder & Mead (1965). Segundo Nash (1990), a essência deste algoritmo é que a função-objetivo é avaliada em cada solução (vértice) do simplex. Em um problema de minimização, o vértice que apresenta maior valor de função-objetivo é substituído por uma nova solução com valor da função-objetivo menor. Existem quatro operações que são realizadas em um simplex: reflexão, expansão, redução e contração.

O simplex inicial move-se, expande-se e contrai-se, de tal maneira a adaptar-se ao panorama da função e, finalmente, aproximar-se do ótimo. Para determinar a transformação apropriada, o método usa só a ordem relativa entre os desempenhos (valor da função a ser otimizada) do ponto considerado. Depois de cada transformação, a solução menos adequada é substituída pela melhor das já existentes. Deste modo, o algoritmo sempre força a convergência em uma determinada seqüência de iterações. De forma a operar-se o simplex é necessário ordenar os pontos de mais alto valor (melhor solução na iteração atual para o problema em questão), x_H, o próximo mais alto, x_N, e o mais baixo, x_L. Assim os valores da função-objetivo, J, associados às soluções do problema obedecem a relação:

$$J(x_H) \geq J(x_N) \geq J(x_i) \geq J(x_L) \quad (7)$$

para todo $i \neq H$, N ou L, onde x são vetores. A figura 3 ilustra a situação do simplex, para um politopo com três lados.

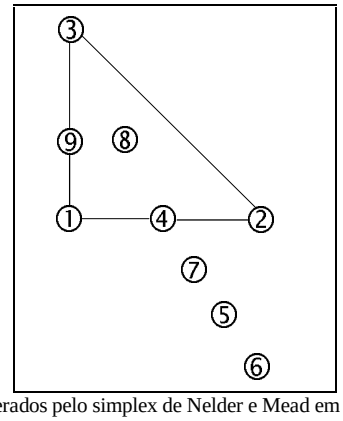

Figura 3. Pontos gerados pelo simplex de Nelder e Mead em duas dimensões (Nash, 1990).

Na figura 3 adota-se as seguintes convenções:
(i) o ponto ①, x_L, é o vértice inferior no simplex;
(ii) o ponto ②, x_N, é o outro vértice, diferente do superior;
(iii) o ponto ③, x_H, é o vértice superior;
(iv) o ponto ④, x_C, é o centróide de todos os vértices, exceto ③(x_H), isto é,

$$x_C = \left. \sum_{\substack{j=1 \\ j \neq H}}^{nptos+1} x_j \right/ nptos \quad (8)$$

(v) o ponto ⑤, x_R, é a reflexão de ③(x_H) através de ④(x_C);
(vi) o ponto ⑥, x_E, é o resultado da extensão da linha (④,⑤), isto é, (x_C, x_R), com o comprimento regido por um fator de expansão;

(vii) o ponto ⑦ é o resultado da redução da linha (④,⑤), que ocorre quando x_R é inferior a x_H, mas superior a x_N;
(viii) o ponto ⑧ é o resultado da redução da linha (④,③), isto é, (x_C, x_H);
(ix) o ponto ⑨ é gerado por uma contração geral do simplex, realizada pelos vértices ①, ② e ③ através de ①(x_L).

A operação de reflexão reflete ③(x_H) através de ④(x_C) usando um fator de reflexão α_R , para a determinação do ponto ⑤, ou seja,

$$x_R = x_C + \alpha_R(x_C - x_H) \tag{9}$$

Se $J(x_R)$ é menor que $J(x_L)$, um novo ponto mais inferior é determinado, e o simplex pode ser expandido estendendo a linha (x_R , x_C) para obter-se o ponto ⑥, ou seja,

$$x_E = x_R + (\gamma_E - 1)(x_R - x_C) \tag{10}$$

onde γ_E é o fator de expansão, é maior que a unidade, caso contrário, x_E representa uma contração. Se $J(x_E)<J(x_R)$ então x_H é repassado por x_E e o procedimento repetido encontrando um novo ponto superior e um novo centróide de *nptos*, x_C. Em outro caso, x_R é o novo ponto inferior e é repassado por x_H. Quando x_R não é o ponto inferior, mas é menor que x_N, tem-se

$$J(x_L) \le J(x_R) \le J(x_N) \tag{11}$$

e x_H é repassado por x_R e o procedimento é repetido. A situação permanecendo, tem-se $J(x_R)$ no mínimo maior que $J(x_N)$ e deve-se reduzir o simplex. Existem duas possibilidades:

(i) Se $J(x_N) \le J(x_R) < J(x_H)$ então a redução é realizada repassando x_H por x_R, e determina-se um novo vértice entre x_C e x_R (atualmente x_H). Esta é uma redução no lado da reflexão (posição inferior).
(ii) Se $J(x_R) > J(x_H)$ a redução é feita determinando-se um novo vértice entre x_C e x_H (posição superior).

Em um ou outro dos casos mencionados a redução é controlada pelo fator β_{NM} com valor entre 0 e 1. Visto que o caso (a) repassa x_H por x_R, a mesma fórmula aplicada para o novo ponto x_S (S denota que o simplex é menor) em ambos os casos. O x_H é utilizado para denotar-se ambos x_R e x_H, desde que no caso (i) x_R tornou-se o novo ponto superior no simplex, ou seja,

$$x_S = x_C + \beta_{NM}(x_H - x_C) \tag{12}$$

O novo ponto x_S repassa o ponto atual x_H, que no caso (i) é, de fato, x_R, a menos que

$$J(x_S) > \text{mínimo } \{J(x_H), J(x_R)\} \tag{13}$$

A substituição de x_H por x_R no caso (i) pode, em uma implementação, medir que este mínimo já foi salvo com seu ponto associado. Quando a equação (13) é satisfeita uma redução dá um ponto maior que $J(x_N)$, assim uma contração geral do simplex sobre o ponto inferior x_L, é sugerida. Ou seja,

$$A_i' = x_L + \beta_{NM}'(x_i - x_L) \tag{14}$$

para todo $i \neq L$. Em aritmética exata, a equação (14) é aceitável para todos os pontos, contudo Nash (1990) faz alguns testes omitindo $i=L$. Alguns cuidados devem ser observados pois o ponto x_L pode ser alterado nas operações da equação (14).

Diferentes fatores de contração β_{NM} e β_{NM}' podem ser utilizados. Na prática estes fatores, assim como os fatores α_R e γ_E, podem ser escolhidos para tentar um aprimoramento da taxa de convergência deste procedimento. Segundo Nelder & Mead (1965), uma escolha adequada é

$$\alpha_R = 1; \gamma_E = 2; \beta_{NM} = \beta'_{NM} = 0{,}5 \tag{15}$$

O método simplex é facilmente combinado com os *AEs*. Recentemente, na literatura foram apresentadas algumas abordagens de algoritmos genéticos com método simplex em Yen *et al.* (1998) e Tsutsui *et al.* (1999). A hibridização da *PE* com o método simplex é realizada para que a *PE* seja executada da forma usual. Entretanto, a cada geração, após a aplicação do operador de mutação, aplica-se o método simplex a *x*% dos melhor(es) indivíduo(s) da população na tentativa de aprimorar o *fitness* do indivíduo sob avaliação. Neste caso, o método simplex é utilizado como uma técnica de busca local da *PE*. Esta combinação é usualmente referida como uma *PE* com mecanismo de subida de encosta, *PE* com aprendizado Lamarckiano ou uma forma de algoritmo memético.

3 Aplicação da metodologia em um estudo de caso em otimização não-linear

Os algoritmos de otimização têm recebido recente atenção para problemas de otimização estrutural e mecânica. A seguir é apresentado um estudo de caso no projeto de um vaso de pressão. O diagrama esquemático do vaso de pressão, apresentado na figura 4, foi introduzido por Sandgren (1990).

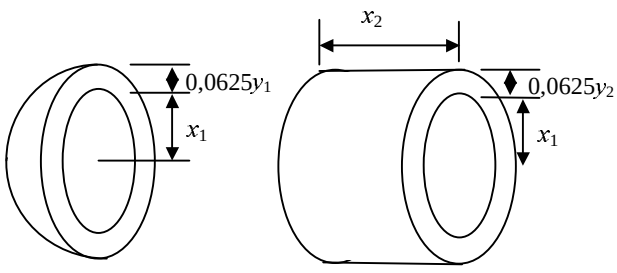

Figura 4. Projeto de um vaso de pressão.

As variáveis de projeto são as dimensões requeridas para o vaso, onde

$$(x, y) = (x_1, x_2, y_1, y_2) \tag{16}$$

A função-objetivo é uma combinação de custos de material, modelagem e soldagem do vaso de

pressão. As restrições são um conjunto de equações que estão de acordo com o código da *ASME* (*American Society of Mechanical Engineers*). O problema de otimização mista (parâmetros inteiros e de ponto flutuante) é expresso como:

$$\min_{x,y} f(x,y) = 0{,}6224(0{,}0625y_1)x_1x_2 + 1{,}7781(0{,}0625y_2)x_1^2 + 3{,}1661(0{,}0625y_1)^2x_2 + 19{,}84(0{,}0625y_1)^2x_1 \quad (17)$$

e sujeito as restrições:

$$g_1(x,y) = 0{,}0193x_1 - 0{,}0625y_1 \le 0 \quad (18)$$

$$g_2(x,y) = 0{,}00954x_1 - 0{,}0625y_2 \le 0 \quad (19)$$

$$g_3(x,y) = 750x1728 - \pi x_1^2x_2 - \tfrac{4}{3}\pi x_1^3 \le 0 \quad (20)$$

$$g_4(x,y) = x_2 - 240 \le 0 \quad (21)$$

As soluções obtidas pelo algoritmo memético usando *simulated annealing* e algoritmo memético usando método simplex são apresentadas na tabela 1 e 2, respectivamente, onde o tamanho da população dos métodos *SA* e simplex indica o número de indivíduos da população avaliados a cada geração. A probabilidades de mutação e cruzamento utilizados no *AG* são 0,1 e 0,8, respectivamente.

Tabela 1: Melhor resultado obtido com *AG+PE hidrida com simulated annealing* (total de experimentos realizados: 50).

tamanho da população			solução para o problema				
AG	*PE*	*SA*	x_1	x_2	y_1	y_2	*f(x,y)*
20	50	10	48,555	110,235	15	8	6372,718
20	100	10	48,576	110,053	15	8	6370,707
20	30	10	48,576	110,058	15	8	6370,705

Tabela 2: Melhor resultado obtido com *AG + PE hibrida com método simplex* (total de experimentos realizados: 50).

tamanho da população			solução para o problema				
AG	*PE*	*NM*	x_1	x_2	y_1	y_2	*f(x,y)*
20	50	10	48,576	110,058	15	8	6370,715
20	100	10	48,576	110,057	15	8	6370,704
20	30	10	48,576	110,056	15	8	6370,703

As soluções obtidas forma similares uma da outra para os algoritmos meméticos propostos. Este aspecto demonstra a robustez das metodologias propostas.

5 Conclusão e futuros trabalhos

Neste artigo foram apresentados os resultados da aplicação de duas novas abordagens de algoritmos meméticos em problemas de otimização não-linear. O estudo de caso apresentado tratou-se do projeto de um vaso de pressão, onde duas variáveis contínuas e duas discretas foram otimizadas. Os resultados foram promissores e mostraram a eficiência das abordagens propostas. As abordagens tratadas foram: (i) *AG* canônico (otimização de parâmetros inteiros) em conjunto com *PE* híbrida com um algoritmo de *SA* (otimização de parâmetros de ponto flutuante), e (ii) *AG* canônico (otimização de parâmetros inteiros) em conjunto com *PE* híbrida com o método simplex de Nelder e Mead (otimização de parâmetros de ponto flutuante).

As perspectivas de futuros trabalhos estão direcionadas para um estudo comparativo e análise de convergência de diversas configurações de algoritmos meméticos em problemas de otimização com múltiplos objetivos.

Referências bibliográficas

Chellapilla, K. (1998). Combining mutation operators in evolutionary programming, *IEEE Trans. on Evolutionary Computation*,**2**(3):91-96.

Coelho, L.S. & A.A.R. Coelho (1999). Algoritmos evolutivos em identificação e controle de processos: uma visão integrada e perspectivas. *SBA Controle & Automação*, **10**(1): 13-30.

Dawkins, R. (1976). *The selfish gene*, Oxford University Press, Oxford, UK.

Kirkpatrick, S.; C.D. Gelatt & M.P. Vecchi (1983). Optimization by simulated annealing, *Science*, **220**: 45-54.

Moscato, P. & M.G. Norman (1992). A 'memetic' approach for the traveling salesman problem — implementation of acomputational ecology for combinatorial optimisation on message-passing systems, *Int. Conf. on Parallel Comp. and Transputer Applications*, IOS Press.

Nash, J.C. (1990). *Compact numerical methods for computers: linear algebra and function minimisation*. 2nd edition. Adam Hilger: NY.

Nelder, J.A. & R. Mead (1965). A simplex method for function minimisation, *Computer Journal*, **7**: 308-313.

Sandgren, E. (1990). Nonlinear integer and discrete programming in mechanical design, *ASME J. Mechanical Design*, **112**: 223-229.

Sun, R.-L. (1995). *Evolving population-based search algorithms through thermodynamic operation: dynamic system design and integration*, PhD thesis, *ISR*, Univ. of Maryland.

Tsutsui, S.; M. Yamamura & T. Higuchi (1999). Multi-parent recombination with simplex crossover in real coded genetic algorithms, *GECCO*, Orlando, FL, pp. 657-664.

Yao, X. & Y. Liu (1996). Fast evolutionary programming, *5th Annual Conf. on Evol. Programming*, San Diego, CA, pp. 451-460.

Yen, J.; Liao, J.C.; B. Lee & D. Randolph (1998). A hybrid approach to modeling metabolic systems using a genetic algorithm and simplex method, *IEEE Trans. on Systems, Man, and Cybernetics — Part B: Cybernetics*, **28**(2): 173-191.