14° POSMEC - Simpósio do Programa de Pós-Graduação em Engenharia Mecânica

Universidade Federal de Uberlândia
Faculdade de Engenharia Mecânica

PARTICLE SWARM OPTIMIZATION

Felipe Antonio Chegury Viana
Universidade Federal de Uberlândia, Faculdade de Engenharia Mecânica.
Av. João Naves de Ávila 2121 – Campus Santa Mônica – CEP 38400-902, Uberlândia, MG.
fchegury@mecanica.ufu.br

Valder Steffen Júnior
vsteffen@mecanica.ufu.br

Resumo: *Os algoritmos clássicos de otimização vêm sendo amplamente utilizados devido a sua eficiência computacional e sucesso na solução de problemas de projeto em Engenharia. Contudo, tais métodos encontram dificuldades ao se depararem com "mínimos locais". Por sua vez, os métodos de otimização natural requerem maior esforço computacional, mas apresentam vantagens tais como: fácil implementação, robustez e não requerem continuidade na definição do problema. Neste trabalho é apresentado o algoritmo conhecido como Particle Swarm Optimization, um algoritmo baseado no comportamento social de aves. A busca por alimento ou pelo ninho e a interação entre os pássaros ao longo do vôo são modelados como um mecanismo de otimização. Desta forma, a área sobrevoada é equivalente ao espaço de projeto e encontrar o local com comida ou o ninho é semelhante a encontrar o ótimo. O algoritmo é baseado em um modelo simplificado da teoria de enxames (swarm theory), através da qual os pássaros ou partículas fazem uso de suas experiências e da experiência do próprio bando para encontrarem o ninho ou alimento. Do ponto de vista sócio-cognitivo, isto quer dizer que a mente e, por conseqüência, a inteligência, são atributos sociais. Os fundamentos teóricos, o algoritmo básico e algumas aplicações são apresentados, incluindo a otimização de funções matemáticas e solução de problemas inversos de identificação de forças em estruturas mecânicas. As aplicações evidenciam a capacidade do algoritmo na solução de diferentes problemas, bem como salientam a habilidade de trabalhar com variáveis discretas e contínuas simultaneamente.*

Palavras-chave: *Otimização, Particle Swarm Optimization, Problemas Inversos.*

1. INTRODUÇÃO

Engenheiros e cientistas estão constantemente procurando por técnicas que permitam encontrar soluções ótimas para os mais diversos problemas. De forma geral, a tarefa de otimização envolve vários componentes: o espaço de projeto (ou de busca), onde são consideradas todas as possibilidades de solução de um determinado problema; a função objetivo (também chamada de função custo, função de avaliação ou critério de desempenho), que representa uma maneira de avaliar os elementos do espaço de projeto; as restrições, que delimitam o espaço de projeto e demarcam a área onde a busca pode ocorrer e quais as limitações e penalidades a função objetivo sofrerá; e o otimizador, ou seja, o algoritmo que irá fornecer a resposta ao problema de otimização. Com freqüência, a chave para resolver com sucesso o problema de otimização está na escolha do método que melhor irá se adequar à solução do problema.

Os algoritmos de otimização podem ser classificados quanto ao tipo de informação que necessitam para resolver o problema (Vanderplaats, 1999). Desta forma:

- Métodos de ordem zero: não usam informação do gradiente.
- Métodos de primeira-ordem: demandam o cálculo do gradiente da função objetivo.

- Métodos de segunda-ordem: requerem uma expansão de segunda ordem da função objetivo.

Uma outra classificação, ilustrada pela Figura 1, é quanto ao paradigma que o método segue. Esta abordagem pode ser clássica, fundamentada em princípios do Cálculo (Vanderplaats, 1999), ou, alternativamente, a abordagem pode ser inspirada em fenômenos naturais, físicos, químicos ou biológicos, originando a chamada otimização natural (Haupt and Haupt, 1998).

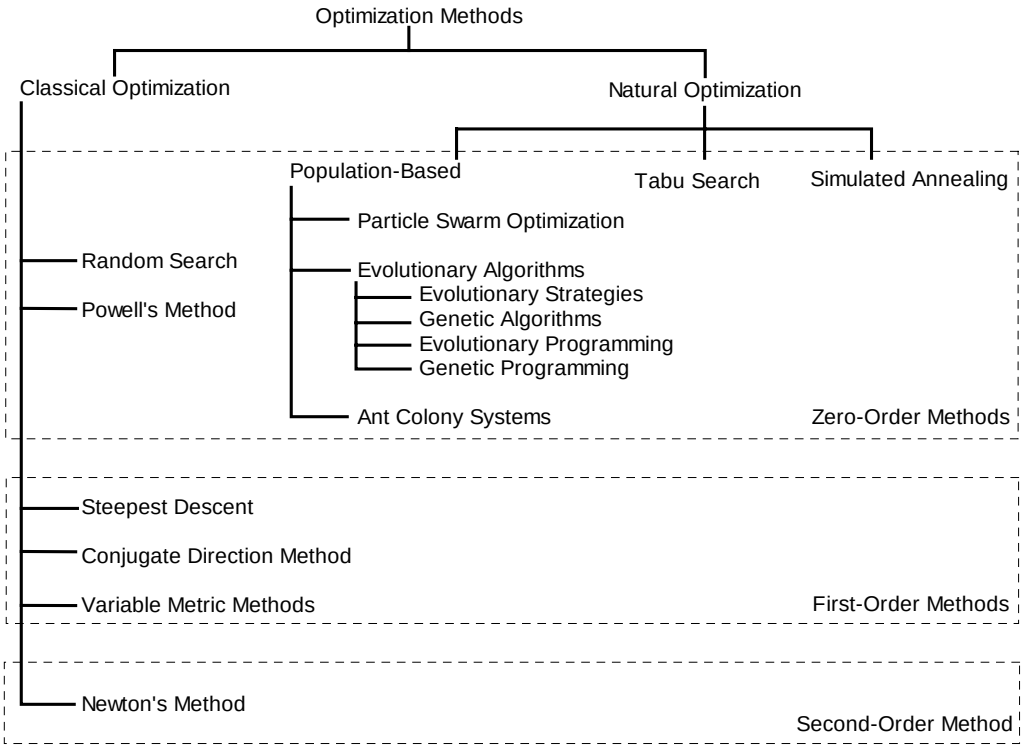

Figura 1: Métodos de Otimização (adaptado de Rasmussen, 2002).

Métodos clássicos possuem uma grande vantagem, qual seja o baixo número de avaliações da função objetivo, o que faz com que tenham convergência rápida. Contudo, estes métodos têm uma inabilidade em lidar com mínimos locais. Como estes métodos trabalham com um único ponto do espaço de projeto e com informações sobre os gradientes, ao se depararem com mínimos locais estes métodos não conseguem avançar na busca, convergindo prematuramente, sem encontrar o mínimo global.

Nos métodos de otimização natural, a função objetivo é avaliada várias vezes, sendo possível trabalhar com vários pontos ao mesmo tempo em uma iteração. Isto eleva o custo computacional destes métodos. Entretanto, isto é compensado pela menor chance que estes métodos têm de se deixarem prender em mínimos locais. Há claramente uma relação de compromisso estabelecida.

2. MÉTODOS NATURAIS DE OTIMIZAÇÃO

De forma geral, os métodos de otimização natural requerem maior esforço computacional quando comparados aos métodos clássicos, mas apresentam vantagens tais como: fácil implementação, robustez e não requerem continuidade na definição do problema (Verter, 2002).

Como exemplo destes métodos podem ser citados os Algoritmos Genéticos. Os Algoritmos Genéticos trabalham técnicas de computação evolutiva, as quais modelam a evolução das espécies proposta por Darwin e operando sobre uma população de candidatos (possíveis soluções). A idéia é que a evolução da população faça com que a formação dos cromossomos dos indivíduos caminhe para o ótimo, à medida que aumenta sua função de adaptação (*fitness*)

Para melhor situar os algoritmos de otimização natural, uma classificação e discussão geral sobre eles é encontrada em Rasmussen, 2002. Esta referência aborda algoritmos adaptativos e propõe a classificação da Figura 2.

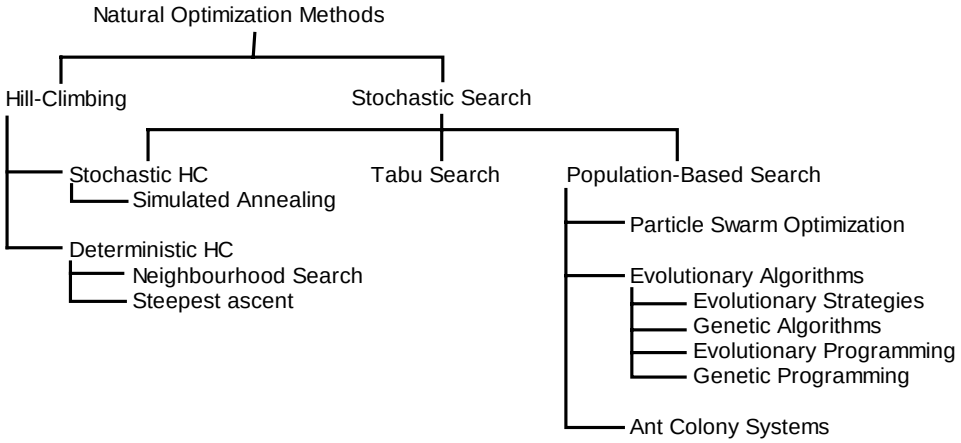

Figura 2: Classificação de métodos naturais de otimização (adaptado de Rasmussen, 2002).

3. OTIMIZAÇÃO POR ENXAME DE PONTOS (*PARTICLE SWARM OPTIMIZATION*)

3.1 Comportamento Social de Enxames Aplicado à Otimização

PSO foi introduzido por James Kennedy e Russell Eberhart em 1995 e emergiu de experiências com algoritmos que modelam o "comportamento social" observado em muitas espécies de pássaros (Pomeroy, 2003). Dentre vários modelos existentes, Kennedy e Eberhart se interessaram particularmente pelo modelo desenvolvido pelo biólogo Frank Heppner.

Imagine a seguinte situação: um grupo de pássaros está aleatoriamente procurando por comida, ou um lugar que lhes sirva como ninho, em uma certa região. Além disso, haveria somente um lugar com comida, ou ninho, em toda região e os pássaros não sabem, *a priori*, onde este lugar está. Então, qual é a melhor estratégia para procurá-lo? A mais efetiva é a de seguir o pássaro que está mais próximo da comida ou do descanso.

Os pássaros de Heppner apresentam as mesmas características de outros modelos, mas acrescentam algo diferente: eles são atraídos para um lugar com comida ou ninho. Em simulações, eles podem começar voando sem nenhuma orientação particular e então, espontaneamente, eles formam bandos até que um ou mais pássaros voam sobre o ninho. Através de regras simples que os pássaros usam para se movimentarem, um pássaro sai do bando para pousar no ninho e acaba atraindo os pássaros mais próximos.

A maneira pela qual um pássaro que encontra o local desejado atrai seus vizinhos aumenta a chance de eles também o encontrarem. Do ponto de vista sócio-cognitivo, isto quer dizer que a mente e por conseqüência a inteligência, são sociais. A essência do ponto de vista da "mente social" é que os indivíduos aprendem principalmente com o sucesso de seus vizinhos.

Desta forma, é necessário um ajuste entre as capacidades de procurar por uma boa solução (exploração - *exploration*) e a de tirar proveito de algo para seu próprio sucesso (proveito - *exploitation*). Se houver pouca exploração a resposta irá convergir para a primeira boa solução encontrada. Se houver pouco proveito, a resposta ótima pode nunca ser encontrada. Em outras palavras, os dois tipos de comportamento que devem ser balanceados são a individualidade e a socialização.

3.2 O algortimo *Particle Swarm Optimization*

O algoritmo é baseado em um modelo simplificado da teoria de enxames (*swarm theory*). Os pássaros (no algoritmo, chamados de partículas) fazem uso de suas experiências e da experiência do próprio bando para encontrarem um local de descanso ou fonte de comida. PSO faz uso de um vetor de velocidades e um vetor de posição para modelar o comportamento das partículas.

Assim, a posição de cada partícula é atualizada considerando a sua velocidade atual, o conhecimento adquirido pela partícula e o conhecimento adquirido pelo bando. O fluxograma da Figura 3 traz um esboço do algoritmo (Rojas et al, 2004):

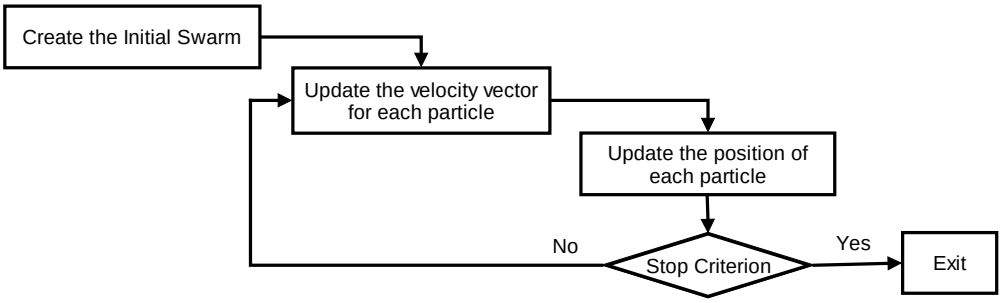

Figura 3: Fluxograma para o algoritmo PSO básico.

A posição das partículas é atualizada segundo a equação:

$$x_{k+1}^i = x_k^i + v_{k+1}^i \Delta t \tag{1}$$

onde x_{k+1}^i representa a posição de cada partícula *i* na iteração *k+1*, v_{k+1}^i representa o vetor de velocidade e Δt corresponde ao passo de tempo.

O vetor de velocidade é atualizado segundo a equação:

$$v_{k+1}^i = w v_k^i + c_1 r_1 \frac{(p^i - x_k^i)}{\Delta t} + c_2 r_2 \frac{(p_k^s - x_k^i)}{\Delta t} \tag{2}$$

onde r_1 e r_2 são números aleatórios entre 0 e 1, p^i é a melhor posição encontrada pela partícula *i* e p_k^s é a melhor posição do bando na iteração *k*.

Existem três parâmetros dependentes do problema, a inércia da partícula (*w*), e os dois parâmetros de confiança c_1 e c_2. A inércia controla a capacidade de exploração do algoritmo, ou seja, um valor alto facilita um comportamento mais global, enquanto um valor baixo facilita um comportamento mais local (Venter and Sobieszczanski-Sobieski, 2002). Os parâmetros de confiança indicam o quanto uma partícula confia em si (c_1), e no bando (c_2). A Figura 4 ilustra a aplicação da equação anterior, ao considerar duas partículas voando em um espaço de projeto bidimensional.

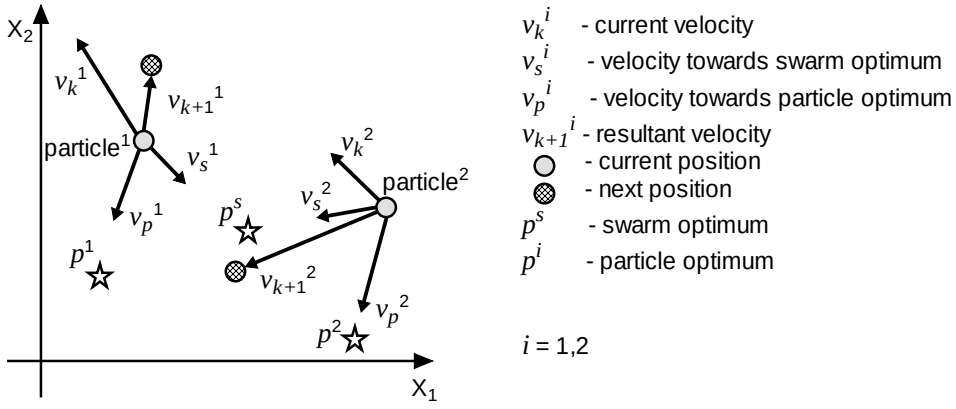

Figura 4: Vetor de velocidades em ação.

3.3 Enxame Inicial

O enxame inicial é geralmente criado com as partículas distribuídas aleatoriamente sobre o espaço de projeto, cada uma com um vetor de velocidade aleatório inicial. A posição e o vetor de velocidade iniciais são obtidos pelas seguintes equações:

$$x_0^i = x_{\min} + r_1(x_{\max} - x_{\min}) \tag{3}$$

$$v_0^i = \frac{x_{\min} + r_2(x_{\max} - x_{\min})}{\Delta t} \tag{4}$$

Nas equações acima r_1 e r_2 são números aleatórios entre 0 e 1, $x_{\min}$ é o vetor de limites inferiores (*lower bounds*) e $x_{\max}$ é o vetor de limites superiores (*upper bounds*) para as variáveis de projeto.

3.4 Parâmetros do Algoritmo

A fórmula usada para a atualização do vetor de velocidade, Equação (2), contém alguns parâmetros que são ajustados de acordo com o problema. São eles os parâmetros de confiança e a inércia. Os parâmetros de confiança devem ser selecionados de forma a balancear a influência do conhecimento adquirido pela partícula e aquele adquirido pelo enxame. A inércia é ajustada para indicar o quanto da velocidade corrente irá permanecer na próxima iteração. A literatura propõe que sejam usados $c_1 = c_2 = 2$ e, para a inércia, valores no intervalo $0.8 < w < 1.4$. Adicionalmente, os parâmetros de confiança podem ser selecionados para valores diferentes, geralmente satisfazendo $c_1 + c_2 = 4$. Para a inércia, um processo de ajuste dinâmico do valor de *w* é proposto por Venter and Sobieszczanski-Sobieski (2002). Segundo estes autores, isto resulta em: 1) uma convergência mais rápida do algoritmo e; 2) a escolha do valor de *w* mais livre ou até sem a interação do usuário.

A inércia é atualizada seguindo a equação:

$$w_{new} = f_w w_{old} \tag{5}$$

onde f_w é uma constante entre 0 e 1. Uma inércia inicial de $w_0 = 1.4$ e $f_w = 0.975$ são usadas no decorrer deste trabalho.

O valor de w não é ajustado em cada iteração . Um coeficiente de variação (CV) dos valores da função objetivo para um subconjunto das melhores partículas é monitorado. Se o CV fica abaixo de um limiar, é assumido que o algoritmo está convergindo para uma solução ótima (Venter and Sobieszczanski-Sobieski, 2002); neste caso, a massa é ajustada seguindo a Equação (5). A equação para o CV é:

$$CV = \frac{StdDev}{Mean} \tag{6}$$

onde $StdDev$ é o desvio padrão e $Mean$ é a média da função objetivo, ou dos vetores de posição, de um conjunto pré-estabelecido de partículas. Neste trabalho um subconjunto de 20% das partículas será monitorado e um limiar de 1.0 é adotado para CV .

3.5 Otimização com Restrições

O PSO, assim como os Algoritmos Genéticos, não trabalha diretamente com restrições. Contudo, através de uma formulação apropriada do problema de otimização, pode-se fazer com que o algoritmo manipule restrições. Uma maneira bastante conhecida é trabalhar com funções de penalidade quadrática estendida (*quadratic extended penalty function*), (Vanderplaats, 1999).

Desta maneira, é criada uma função pseudo-objetivo definida como:

$$\tilde{f}(x) = f(x) + \alpha \sum_{i=1}^{m} \max[0, g_i(x)]^2 \tag{7}$$

sendo $f(x)$ a função objetivo original, α o parâmetro de penalidade (de ordem variável segundo o tipo de problema), $g_i(x)$ o conjunto de todas as restrições (com violações para $g_i(x) > 0$) e $\tilde{f}(x)$ a nova função objetivo.

3.6 Manipulando Violações

Nos problemas de otimização envolvendo restrições, partículas que violam alguma restrição merecem atenção especial. O tratamento da violação começa pela novo cálculo do vetor de velocidade, avaliado segundo a equação:

$$v_{k+1}^i = c_1 r_1 \frac{(p^i - x_k^i)}{\Delta t} + c_2 r_2 \frac{(p_k^s - x_k^i)}{\Delta t} \tag{8}$$

Logo após, o vetor de posição é atualizado de acordo com a Equação (1), onde x_k^i é o vetor de posição antes da violação.

Assim, diferentemente da formulação inicial, a Equação (8) não considera a informação do vetor de velocidade na iteração anterior para o cálculo do vetor velocidade da iteração atual. Isto porque a partícula estaria "voando" rumo a uma violação. Desprezando esta informação, resta apenas p^i, a melhor posição encontrada pela partícula *i* e p_k^s, a melhor posição do bando na iteração *k*. Segundo Venter and Sobieszczanski-Sobieski (2002), na maioria dos casos este novo vetor de velocidades apontará para uma região realizável e a partícula sai da restrição.

3.7 Variáveis de Projeto Discretas/Inteiras

Diferentemente de Algoritmos Genéticos, inicialmente propostos para variáveis discretas, PSO é inerentemente um algoritmo contínuo. Contudo, o algoritmo tem um potencial considerável em resolver problemas discretos e/ou funções ou variáveis descontínuas. Neste trabalho, uma pequena modificação, dentre várias possíveis, é introduzida ao algoritmo básico de PSO para resolver problemas com variáveis discretas. A técnica é direta: a posição de cada partícula é arredondada para o valor inteiro mais próximo logo após a aplicação da Equação (1) ou da Equação (3). Este método, ainda que simples, tem demonstrado eficiência nos problemas testados.

3.8 Operador de Perturbação (*Craziness Operator*)

Para evitar convergência prematura do algoritmo PSO, uma aleatoriedade adicional pode ser introduzida usando o operador de perturbação (*craziness operator*). Este operador atua como o operador de mutação nos Algoritmos Genéticos. O operador de perturbação altera tanto o vetor de posição quanto o vetor de velocidade da partícula afetada. A posição da partícula é alterada aleatoriamente, enquanto o vetor de velocidade é recalculado segundo a equação:

$$v_{k+1}^i = c_1 r_1 \frac{(p^i - x_k^i)}{\Delta t} \tag{9}$$

O operador é aplicado ao fim de cada iteração e as partículas a serem alteradas são identificadas pelo mesmo coeficiente de variação (*CV*) definido anteriormente. Se o valor do *CV* estiver abaixo de um limiar é assumido que o enxame está se tornado muito uniforme. Neste caso, partículas que estão longe do centro do enxame são identificadas, usando o desvio padrão das coordenadas de cada partícula. Partículas que estão localizadas a mais de duas vezes o desvio padrão do centro do enxame são sujeitas ao operador de perturbação.

Venter and Sobieszczanski-Sobieski (2002) afirmam que não há um consenso quanto a eficiência deste operador. Segundo esta referência, Kennedy e Eberhart concluíram que este operador pode não ser necessário, enquanto Fourie e Groenwold reintroduziram o operador em suas aplicações. Desta maneira, fica a cargo do usuário a decisão de utilizar ou não este operador.

4. APLICAÇÕES

Para realização de testes e posteriores aplicações em problemas reais de otimização, foi escrito um código na forma de um *toolbox* para MATLAB®.

4.1 Aplicação em Funções Multimodais

Este tipo de problema é especialmente interessante por possibilitar a análise de operadores, critérios de parada, manipulação de restrições, etc. Neste trabalho, a aplicação tem por objetivo minimizar da função dada por:

$$f(x) = -\left|x^3.\sin(2\pi x)\right|, \qquad 0 \le x \le 10 \tag{10}$$

A Tabela 1 traz as configurações do algoritmo para a aplicação.

Tabela 1: Configuração dos parâmetros do PSO para a aplicação em funções multimodais.

Número de partículas	*w*	*Self trust* (c_1)	*Swarm trust* (c_2)	Δt	Máximo de iterações
20	1.4	1.5	2.5	1	75

Os resultados podem ser visualizados na Figura 5. As Figura 5-(a) e (b) mostram o gráfico da função objetivo e a distribuição inicial e final do enxame em uma das simulações realizadas; o ótimo encontrado foi $x_{optm} = 9.758$. Na Figura 5-(c) tem-se o resultado para 1000 simulações. E finalmente, na Figura 5-(d) tem-se uma análise estatística para os resultados.

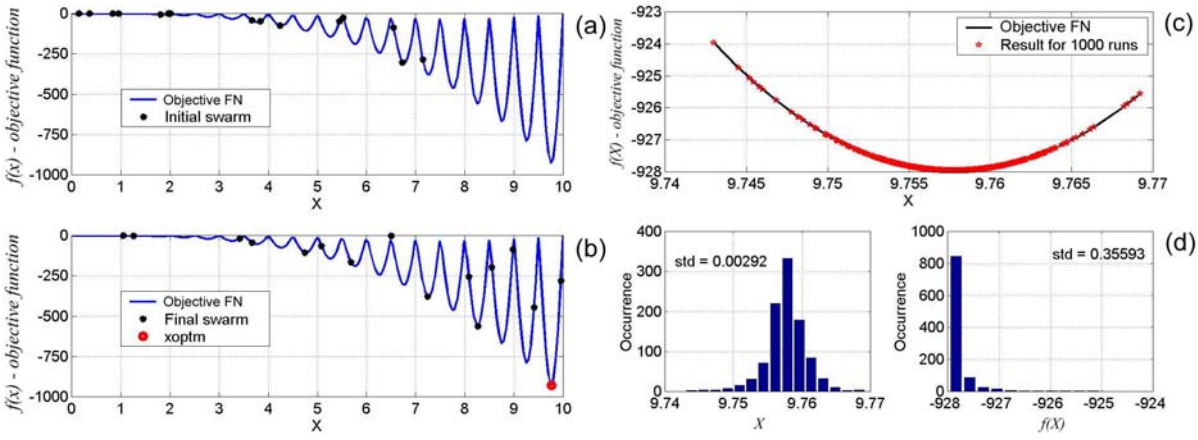

Figura 5: Resultados para a primeira aplicação.

É possível perceber uma baixa dispersão nos resultados ao longo das 1000 simulações bem como a robustez do algoritmo, ao fugir dos mínimos locais.

4.2 Aplicação em Engenharia

Em diversas aplicações de Engenharia Estrutural é importante conhecer o carregamento externo a que estão submetidas as estruturas em condições reais, objetivando avaliar o nível de segurança, verificar as considerações adotadas no projeto e promover o redimensionamento de componentes.

Considerando a influência exercida pelo carregamento externo sobre as respostas dinâmicas através do efeito conhecido por enrijecimento por tensão, é possível, através de um procedimento inverso, obter informações acerca dos níveis e distribuições de cargas a partir das respostas dinâmicas medidas da estrutura (Rojas, 2004).

Uma vez que nos problemas inversos a unicidade da solução não pode ser garantida, verifica-se que as técnicas clássicas de otimização ficam comprometidas em virtude da presença de mínimos locais no espaço de projeto. Neste sentido, é proposta uma metodologia para a determinação das cargas externas em estruturas através de uma técnica de problemas inversos, a partir das respostas dinâmicas dos sistemas e usando PSO seguido por um algoritmo clássico baseado na programação quadrática seqüencial (Sequential Quadratic Programming - SQP), assim como pode ser visto em Rojas (2004) e Rojas et al (2004).

Neste trabalho, foram realizadas simulações que objetivam identificar a magnitude, a posição e a direção de um carregamento aplicado em uma estrutura bidimensional, ilustrada na Figura 6.

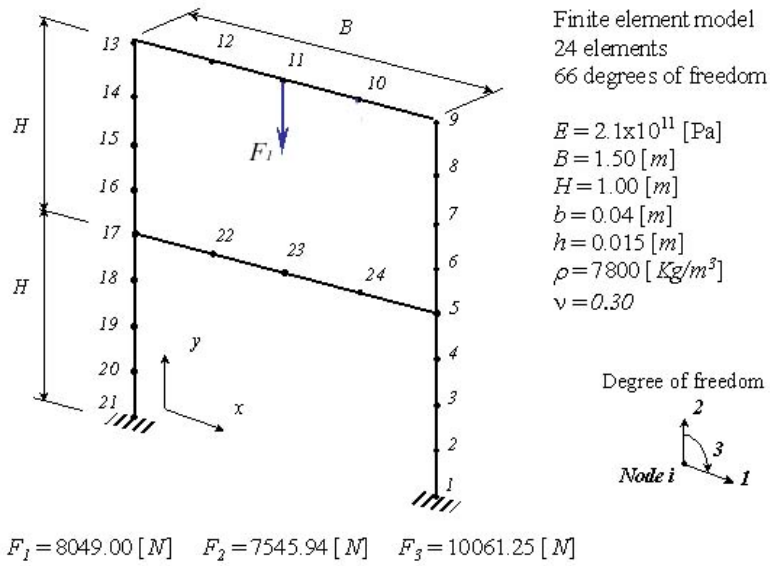

Figura 6: Modelo de Elementos Finitos para um pórtico bidimensional.

A partir do modelo de elementos finitos, é possível obter as freqüências naturais da estrutura submetida ou não a qualquer carregamento. Desta forma um experimento computacional é realizado, originando o sistema que se deseja identificar. Por fim, constrói-se uma função objetivo que caracteriza as diferenças entre o sistema a ser identificado e todas as outras combinações indesejadas dentro do espaço de busca. Matematicamente a função objetivo pode ser escrita como:

$$J(\boldsymbol{p}) = \sum_{i=1}^{m} \frac{\left|\omega_i^{(m)}(\boldsymbol{p}) - \omega_i^{(e)}\right|}{\omega_i^{(e)}} \tag{11}$$

sendo:

- m o número de freqüências naturais usadas,
- $\boldsymbol{p}$ o vetor de parâmetros para o carregamento (a ser identificado),
- $\omega_i^{(m)}(\boldsymbol{p})$ as freqüências naturais calculadas a partir do modelo de elementos finitos e
- $\omega_i^{(e)}(\boldsymbol{p})$ as freqüências naturais do experimento (estrutura com carregamento).

Os seguintes cenários são estudados: 1) identificação da magnitude de F_1; 2) identificação da magnitude e posição de F_1; 3) identificação da magnitude, posição e direção de F_1. A Tabela 2 mostra as freqüências naturais com e sem os carregamentos propostos para cada cenário.

Tabela 2: Freqüências naturais para o pórtico bidimensional.

Cenário		**Freqüências Naturais [Hz]**					
		1	**2**	**3**	**4**	**5**	**6**
Sem carregamento		4.42	15.07	22.74	28.30	51.85	59.64
1, 2, 3	F_1 **[N]**	3.49	13.20	21.69	28.56	49.89	58.05

A Tabela 3 mostra os resultados da identificação dos vários cenários obtidos. Como em aplicações reais, métodos de identificação devem ser robustos o suficiente para manipular erros experimentais. Assim, foi considerada uma situação na qual os dados experimentais foram corrompidos com 10% de erro aleatoriamente, como mostra a Tabela 3. Na tabela F_i refere-se à magnitude do carregamento e P_i e D_i denotam a posição e a direção de F_i.

Tabela 3: Resultados da identificação usando PSO e PSO + SQP.

			Sem dados corrompidos			**Com dados corrompidos**			
			PSO	PSO + SQP		**PSO**		PSO + SQP	
Cenário		**Exato**	**Ótimo**	**Ótimo**	**Erro [%]**	**Ótimo**	**Erro [%]**	**Ótimo**	**Erro [%]**
1	F_1 **[N]**	8049.0	8039.6	8049.0	$3.7\text{x}10^{-6}$	8826.3	9.7	8881.7	10.3
2	F_1 **[N]**	8049.0	7742.6	8049.0	$7.5\text{x}10^{-6}$	7552.8	6.1	7552.8	12.4
	P_1	11	11	-	0	11	0	-	0
3	F_1 **[N]**	8049.0	8465.3	8049.0	$3.7\text{x}10^{-6}$	7822.2	2.8	7802.6	3.1
	P_1	11	11	-	0	11	0	-	0
	D_1	2	2	-	0	2	0	-	0

5. CONCLUSÕES E TRABALHOS FUTUROS

As simulações realizadas propõem-se a testar o algoritmo e a sua implementação. A partir delas pode-se concluir que o algoritmo PSO se mostrou versátil, operando com sucesso nos casos de otimização de funções multimodais e também nas aplicações de Engenharia. Os resultados obtidos pela análise estatística das buscas em funções multimodais mostram que o algoritmo apresenta robustez e ótima repetibilidade de resultados. Nas aplicações em Engenharia, PSO mostra habilidade na otimização com variáveis puramente discretas e com variáveis discretas e contínuas simultaneamente, particularmente no caso do tratamento de problemas inversos.

Com vistas à divulgação e ao desenvolvimento de métodos naturais de otimização, podem ser sugeridos, como trabalhos futuros, testes com o código desenvolvido na solução de diferentes problemas de Engenharia, aprimoramento do código já disponível e implementando novas rotinas, além de comparar os resultados obtidos com outros algoritmos de otimização.

6. AGRADECIMENTOS

Os autores são gratos à CNPq pelo suporte financeiro oferecido para este trabalho.

7. REFERÊNCIAS

Haupt, R. L. and Haupt, S. E.; “Pratical Genetic Algorithms”, Wiley-Interscience Publication, New York, 1998.

Hu, X., “PSO Tutorial”, http://web.ics.purdue.edu/~hux/tutorials.shtml, [21 Setembro de 2003].

Kennedy, J. and Eberhart, R. C., “Particle Swarm Optimization”, Proceedings of the 1995 IEEE International Conference on Nerual Networks, Perth, Australia, 1995,pp. 1942-1948.

Pomeroy, P., “An Introduction to Particle Swarm Optimization”, http://www.adaptiveview.com, [15 Setembro de 2003].

Rasmussen, T. K., “Improving Particle Swarm Optimization by hybridization of stochastic search heuristics and Self-Organized Criticality”, Doctoral thesis, University of Aarhus, Department of Computer Science, Aarhus C, Denmark, May 2002.

Rojas, J. E. F., “Caracterização do Efeito de Enrijecimento por Tensões e Identificação de Cargas em Estruturas baseada em Respostas Dinâmicas”, Dissertação de Mestrado, Universidade Federal de Uberlândia, MG–Brasil, 2004.

Rojas, J.E., Viana, F.A.C. , Rade, D.A. and Steffen Jr, V., "Force identification of mechanical systems by using particle swarm optimization". In Proceedings of the 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Albany, New York, Aug 30-01 Sept 2004.

Vanderplaats, G. N., “Numerical Optimization Techniques for Engineering Design”, Vanderplaats Research and Development, Inc., 3rd ed., 1999.

Venter, G. and Sobieszczanski-Sobieski, J., “Particle Swarm Optimization”, Proceedings of the 43rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Denver, CO, Vol. AIAA-2002-1235, April 22-25 2002.

PARTICLE SWARM OPTIMIZATION

Felipe Antonio Chegury Viana
Federal University of Uberlândia, School of Mechanical Engineering
2160 João Naves de Ávila Av., Campus Santa Mônica, CEP 38400-902, P.O. Box 593, Uberlândia, Brazil
fchegury@mecanica.ufu.br

Valder Steffen Júnior
vsteffen@mecanica.ufu.br

Abstract: *Classical optimization methods have been widely used in the solution of design problems in Engineering due their computational efficiency and success. However, these methods find difficulties when they search along “local minima”. In the other hand, natural optimization methods require more computational effort, but they have shown advantages such as: easiness of implementation, robustness, they do not require continuity of the functions involved in the optimization problem. In this work it is presented the algorithm known as Particle Swarm Optimization, an algorithm based on the social behavior of birds. The search procedure for food or nest and the interaction among the birds through the flying are modeled as an optimization mechanism. By this way, the flight area is equivalent to the design space and to find food or nest is similar as to find the optimum. The algorithm is based on a simplified model of the swarm theory, in which the birds or particles use their own experience together with the swarm experience in order to find food or nest. In the socio-cognitive viewpoint, this means that the mind and, as a consequence, the intelligence, are social attributes. The theoretical foundations, the basic algorithm and some applications are presented, including the optimization of analytical functions and solution of inverse problems as the one dedicated to force identification of mechanical structures. The applications indicate the capacity of the heuristic techniques to obtain the solution of different problems, as well they point out the ability of dealing with discrete and continuous variables simultaneously.*

Keywords: *Optimization, Particle Swarm Optimization, Inverse Problems.*