X Congresso Brasileiro de Inteligência Computacional (CBIC'2011), 8 a 11 de Novembro de 2011, Fortaleza, Ceará

Um Algoritmo Híbrido Baseado na Mutação Creep para Resolução de Problemas de Otimização com Restrições

Raul da S. Barros, Omar A. C. Cortes, José Daniel P. Filho, Rafael F. Lopes
Instituto Federal de Educação, Ciência e Tecnologia do Maranhão
Av. Getulio Vargas, 04 - Monte Castelo São Luis - MA - Brasil
rauls.sdg@gmail.com, omar@ifma.edu.br, jdanielprf@gmail.com, rafaelf@ifma.edu.br

Resumo – Este artigo apresenta um algoritmo híbrido que combina Otimização em Nuvem de Partículas (PSO) com Algoritmos Genéticos (AG) na resolução de problemas de otimização com restrições. Basicamente o algoritmo é baseado em PSO com mutação genética *Creep*, cujo objetivo é aumentar a variabilidade das possíveis disposições das partículas, ou seja, aumentar a variabilidade dentro da população. A seleção de partículas onde é aplicada a mutação é feita através de uma constante chamada taxa de mutação, que corresponde a uma porcentagem do total de partículas presentes na nuvem, sendo que nas partículas selecionadas, todas as dimensões sofrem mutação. O algoritmo foi aplicado em três funções de benchmarks com restrições e seus resultados são avaliados e comparados com os resultados obtidos pelo PSO canônico. Além disso, os resultados são comparados também com o encontrado na literatura.

Palavras-chave – Nuvem de Partículas, mutação creep, benchmarks, restrições.

Abstract – This paper presents an hybrid algorithm mixing a Particle Swarm Optimization (PSO) algorithm with Genetic Algorithms for solving multi-objective problems. The PSO-based algorithm uses a genetic mutation aiming to improve the position variability of the particles inside the swarm. Particles are selected according to a mutation rate, which correspond to a percentage of the particle undergoing mutation, inside the swarm. All dimensions of a particle are muted in order to improve the variability of the population. This algorithm has been applied to solve three constrained benchmarks. Its results are analyzed and compared with canonical PSO. Our results are compared between other ones with the literature, as well.

Keywords – Particle Swarm, creep mutation, benchmarks, constraints.

1 Introdução

As formas de otimização e busca estocástica, inspirados nos princípios e modelos da evolução biológica natural, têm atraído grande interesse nas últimas décadas, devido principalmente, à sua versatilidade na resolução de problemas complexos, especialmente problemas de engenharia, os quais geralmente estão sujeitos a restrições.

Parte dos princípios oriundos do mundo biológico são baseados na teoria da evolução Darwiniana, onde os algoritmos tentam abstrair e imitar alguns dos mecanismos evolutivos, objetivando solucionar problemas que requerem adaptação, busca e otimização.

O desenvolvimento de modelos computacionais, baseados nos mecanismos da evolução biológica, caracterizam-se pela criação de algoritmos de otimização robustos e sistemas adaptativos. Os Algoritmos Evolucionários (AEs) [1] [2], metodologias da área computação evolucionária ou evolutiva (CE), não são algoritmos computacionais em seu significado usual, mas formam uma classe de métodos regidos por princípios similares.

A Otimização por Nuvem de Partículas (PSO) [3] é uma técnica de computação evolucionária desenvolvida por James Kennedy, um psicólogo social, e por Russell Eberhart, um engenheiro elétrico, em 1995, inspirada na simulação de um sistema social simplificado. A intenção original era simular graficamente o comportamento de um bando de pássaros em vôo com seu movimento localmente aleatório, mas globalmente determinado.

Computacionalmente, os Algoritmos de Nuvem de Partícula ou Enxame de Partículas são uma abstração desse processo natural, onde a procura pela posição mais apta é a busca de uma solução ótima para um problema, sendo o conjunto de possíveis posições das partículas o espaço de busca do problema, e, cada posição ocupada por uma partícula representa uma possível solução para o problema.

O comportamento de cada partícula é baseado na sua experiência anterior e na experiência daqueles outras partículas com os quais ele se relaciona. Similarmente aos algoritmos genéticos, o conjunto das partículas tende a preservar aquelas posições que determinam uma maior aptidão e a descartar as posições de menor aptidão.

A diferença entre um Algoritmo Genético (AG) e a otimização por nuvem de partículas está no fato de que para cada solução potencial é também designada uma velocidade aleatória e, as soluções potenciais, chamadas partículas, voam através do espaço de busca do problema. Cada partícula mantém o rastro de suas coordenadas no espaço de busca, que são associadas com a melhor solução (aptidão) que ela tenha alcançado, sendo que esse valor é chamado de *pbest*.

Além da aptidão, o melhor valor de todos os obtidos por qualquer partícula da população também é armazenado, o que faz com que o algoritmo seja elitista. O princípio dessa metaheurística consiste de, em cada iteração mudar a velocidade, para que as partículas voem em direção de suas posições de ótimo [4].

Embora, PSO seja um algoritmo interessante para a solução de problemas com restrições, hibridiza-lo com outros algoritmos tem mostrado bons resultados, por exemplo, ele pode também ficar preso em ótimos locais. Nesse contexto, neste trabalho aplicamos a mutação dos algoritmos genéticos para alterara a velocidade das partículas em uma nuvem. Esse hibridismo é aplicado na otimização de três problemas com restrições chamados de benchmarks.

Para cumprir seu papel este artigo esta dividido da seguinte forma: a Seção 2 apresenta alguns trabalhos de sucesso na hibridização de algoritmos; a Seção 3 mostra o funcionamento de uma nuvem de partículas tradicional; a Seção 4 introduz o algoritmo híbrido desenvolvido; a Seção 5 ilustra os resultados obtidos; finalmente, na Seção 6 são apresentadas as conclusões deste trabalho.

2 Trabalhos Correlatos

Muitos trabalhos, baseados em algoritmos híbridos tem aparecido. Por exemplo, o trabalho de [5] mistura evolução diferencial e estratégias evolutivas para solucionar algumas funções monomodais e multimodais. Uma mistura entre evolução diferencial, simulated annelaig e teoria do caos e feita no trabalho de 6. PSO e algumas variações baseadas na teoria de autômatos são aplicados na otimização de funções de benchmarks no trabalho de [7]. Enquanto que no trabalho de [8] são usados operadores genéticos em um algoritmo de seleção clonal de modo a melhorar as afinidades dos antígenos na resolução de algumas funções de benchmarks. [9] faz apenas a comparação entre PSO e GA, mostrando que PSO apresenta bons resultados na otimização de parâmetros em uma planta industrual. Por fim, o trabalho de [10] faz o hibridismo entre PSO e Algoritmos Genéticos, usando o método da roleta, realizando cruzamentos e mutações nos indivíduos de uma nuvem na resolução de problemas de otimização com e sem restrições.

Especificamente tratando de mutação aplicada ao PSO, o trabalho de [11] usa a mutação aleatória em algumas dimensões da nuvem de partículas caso a nuvem não evolua, ou, evolua muito pouco dada uma certa quantidade de iterações. Em [12] utiliza-se a mutação BGA [13] para mutar algumas partículas, evitando assim a convergência prematura. Já o trabalho de [14], aplica três tipos diferentes de mutação (Cauchy, Gaussiana e Levy), além de usar uma heurística para escolher automaticamente qual das três utilizar. Em todos esses trabalhos, os algoritmos de PSO com mutação são aplicados à otimização numérica, ou seja, sem restrições.

Sabendo da possível melhora nos resultados dos algoritmos hibridizados resolveu-se testar a técnica de hibridização em um algoritmo de PSO canônico, já que o mesmo é uma boa alternativa para resolução de problemas complexos, sem esquecer que em determinados casos ele se mostra não muito atraente se comparado com outros algoritmos de resolução de problemas com restrições, como por exemplo, os AGs.

Pensando nisso surgiu à idéia de criar um algoritmo de otimização por nuvem de partículas que tivesse uma característica de AGs, mas que não utiliza-se tantas operações quanto o trabalho de [10] [1] e fosse diferente das abordagens usadas em [11], [12] e [14]. . Nesse contexto, utilizou-se somente a mutação Creep [2] presente nos algoritmos genéticos.

3 Otimização por Nuvem de Partícula - PSO

O PSO se diferencia de outras técnicas de computação evolutiva principalmente pelo fato de não utilizar operadores genéticos, como acontece nos AGs. Por não utilizar tais operadores as partículas da nuvem ajustam seu vôo de forma individual, movimentando-se no espaço de busca através de suas experiências e da experiência de seus vizinhos. A partir disso, um conjunto de partículas (nuvem ou enxame) é colocado no espaço de busca com o objetivo de procurar um ótimo local, baseando-se em alguns procedimentos determinísticos. As partículas se comunicam entre si informando os valores da função objetivo em suas respectivas posições locais.

Cada movimento de otimização da partícula é baseada em três parâmetros: fator de sociabilidade, fator de individualidade e velocidade máxima. O algoritmo combina estes parâmetros com um número gerado aleatoriamente para determinar o próximo local da partícula. O objetivo de cada fator é:

- *Fator de sociabilidade* - determinar a atração das partículas para a melhor posição descoberta por qualquer elemento do enxame;
- *Fator de individualidade* - definir a atração da partícula com sua melhor posição já descoberta;
- *Velocidade máxima* - delimitar o movimento da partícula, uma vez que esse é direcional e determinado.

Além destes três fatores, têm-se ainda o número de partículas em uma nuvem, o número de nuvens no espaço solução e os critérios de parada. Cada partícula é tratada como um ponto em um espaço D-dimensional. A i-ésima partícula é representada como $X_i = (X_{i_1}, X_{i_2}, \ldots, X_{i_D})$. A melhor posição prévia (a posição que dá o melhor valor de aptidão) da i-ésima partícula é

[1] A medida que mais operações são adicionadas, mais processamento é necessário.

[2] A mutação Creep consiste em adicionar ou subtrair um pequeno valor a um gene. No caso do PSO, adiciona-se ou subtrai-se um pequeno valor de uma dimensão.

registrada e representada como $P_i = (P_{i_1}, P_{i_2}, \ldots, P_{i_D})$. O índice da melhor partícula entre todas as partículas na população é representado por g. A taxa da mudança de posição (velocidade) para a partícula i é representada como $V_i = (V_{i_1}, V_{i_2}, \ldots, V_{i_D})$. As partículas são manipuladas de acordo com as Equações 1 e 2, onde c_1 e c_2 são duas constantes positivas que correspondem às componentes cognitivas e sociais, respectivamente, r é um número aleatório no intervalo $[0, 1]$ e W é o peso de inércia.

$$V_{id} = W * V_{id} + c_1 * r * (P_{id} - X_{id}) + c_2 * r * (P_{gd} - X_{id}) \quad (1)$$

$$X_{id} = X_{id} + V_{id} \quad (2)$$

A Equação 1 é utilizada para calcular uma nova velocidade para uma partícula, de acordo com a velocidade anterior e as distâncias entre sua posição atual, sua melhor posição e a melhor posição do grupo. Feito isto a partícula muda para uma nova posição de acordo com a Equação 2.

O desempenho de cada partícula é avaliado através de uma função de aptidão pré-definida que é relacionada ao problema a ser resolvido. O peso de inércia W é empregado para controlar o impacto da velocidade anterior na velocidade atual, influenciando assim as habilidades de exploração global e local das partículas. Vale ressaltar que um peso de inércia maior facilita exploração global (a procura de novas áreas), enquanto um peso de inércia menor tende a facilitar exploração local para refinar a área de procura atual. Então a seleção satisfatória do peso de inércia W pode prover um equilíbrio entre habilidades de exploração global e local, e assim pode requerer menos repetições, em média, para encontrar o valor ótimo.

As partículas armazenam os valores de suas coordenadas no espaço do problema de forma individual. Esse valor, chamado de $pbest$, é associado com a melhor solução, sendo na verdade, o quanto a partícula deslocou-se. Outro valor que é determinado pelas partículas é o melhor valor obtido por qualquer partícula vizinha, chamado de $lbest$. Quando uma partícula levar toda a população, ou seja, seus vizinhos, em uma direção é chamado $gbest$.

O número de enxames em um espaço é claramente conhecido como um fator na probabilidade de achar o ótimo, pois, quanto maior o número de partículas em um determinado espaço, mais alta será a probabilidade de achar o ótimo. Porém, reciprocamente, um número maior de partículas resultará no aumento de pontos individuais que serão testados, aumentando assim o tempo de computação. O pseudocódigo para a representação do algoritmo PSO pode ser visto no pseudocódigo a seguir, sendo, m o tamanho da população de partículas, $f(x_i)$ a aptidão da partícula i, P_i a melhor posição encontrada pela partícula i e g a melhor posição encontrada por todas as partículas.

Inicializar a nuvem de partículas
repeat
  for i = 1 to m **do**
    if $f(x_i) < f(p_i)$ **then**
      $p_i = x_i$
      if $f(x_i) < f(g)$ **then**
        $g = x_i$
      end if
    end if
    for j = 1 to n **do**
      $r_1 = rand()$
      $r_2 = rand()$
      $V_{ij} = WV_{ij} + C_1 r_1 (p_i - x_{ij}) + c_2 r_2 (g_j - x_{ij})$
    end for
    $x_i = x_i + v_i$
  end for
until critério de parada atingido

No algoritmo, a nuvem de partículas é lançada inicialmente dentro do espaço de busca, tendo cada partícula as seguintes características:

- Uma posição e uma velocidade;
- Conhecimento de sua posição e o valor da função objetivo para esta posição;
- Conhecimento sobre seus vizinhos (vizinhança): a melhor posição encontrada e o valor da sua função objetivo;
- Armazenamento de sua melhor posição encontrada. Em cada espaço de tempo, o comportamento de uma partícula é determinado dentre três possíveis escolhas: (a) Seguir seu próprio caminho; (b) Seguir para sua melhor posição encontrada; (c) Seguir para a melhor posição encontrada por algum de seus vizinhos.

A condição de parada é ultrapassar o limite de número de iterações pré-definido ou quando não houver mais melhorias. A seguir apresenta-se o algoritmo PSO híbrido.

4 PSO Híbrido (PSO-H)

No PSO tradicional não existe o conceito de indivíduo, mas sim de partícula que se movimenta no espaço de busca. No entanto, percebeu-se que a mutação dos algoritmos genéticos poderia ser utilizada como forma de diversificar ainda mais a população de partículas.

Nos AGs a mutação ocorre na população por inteiro e somente em alguns genes (dimensões), uma vez que uma taxa de mutação baixa tende a prevenir o algoritmo de cair em ótimos locais. No algoritmo híbrido proposto a mutação aplicada no PSO não ocorre em todas as partículas, mas somente em um pequeno percentual da nuvem. Uma vez que a mutação ocorre em todas as dimensões da partícula isso pode ajudar a aumentar a variabilidade entre as partículas, algo como ocorre nas estratégias evolutivas [15]. Nesse contexto, uma taxa de mutação baixa pode prevenir a nuvem de se estagnar em um ótimo local, além de aumentar a possibilidade de se chegar a qualquer ponto do espaço de busca. O que não aconteceria com uma taxa muito alta, pois a busca se tornaria praticamente aleatória.

Como visto na Seção 3, na otimização por PSO o fator que influencia na mudança de posição das partículas é a velocidade (taxa de mudança de posição de uma partícula). Assim sendo, no algoritmo híbrido proposto, a mutação é aplicada sobre a velocidade da mesma, sendo feita da seguinte forma: primeiramente determina-se uma porcentagem da nuvem para sofrer mutação, em seguida, de forma aleatória são escolhidas as n partículas que irão ser modificadas; por fim, cada partícula obtém um novo valor de velocidade que em seguida é aplicado à partícula através Equação 2. Após a mutação, as partículas tem seu valor de aptidão recalculado e o algoritmo prossegue normalmente. O pseudocódigo da mutação é apresentado a seguir.

$z \leftarrow$ porcentagem de partículas da nuvem
for i = 1 to z **do**
  Selecionar aleatoriamente uma partícula da Nuvem
  for j = 1 ate numDimensoesParticula **do**
    $V_i = Rand()//[0,1]$
    $P_i = P_i + V_i$
  end for
end for

Foram pensados diversos meios de atribuir um novo valor para a velocidade da partícula, uma delas foi recalcular a velocidade baseada na Equação 1. Outra forma consistia em gerar um valor aleatório qualquer, porém a maneira que proporcionou os melhores resultados, ou melhor dizendo, as soluções mais próximas do ideal foi a geração de um número aleatório no intervalo $[0.1]$. Em se tratando da taxa de mutação o valor mais satisfatório foi cerca de dez por cento da nuvem.

5 Resultados

Para avaliar o algoritmo foram utilizadas três funções de teste multi-objetivo conforme as Equações 3, 4 e 5, que são chamadas no restante deste trabalho de E_1, E_2 e E_3, respectivamente, sendo que a otimização de um problema com restrições consiste em atender a todas as restrições de forma rígida, ou seja, não se pode atender uma restrição e violar outra.

$$
\begin{aligned}
\text{Min } f(x) &= 5\sum_{d=1}^{4} x_d - 5\sum_{d=1}^{4} x_d^2 - 5\sum_{d=5}^{13} x_d \\
\text{s.t. } g_1(x) &= 2x_1 + 2X_2 + x_{10} + x_{11} - 10 \leq 0 \\
g_2(x) &= 2x_1 + 2X_3 + x_{10} + x_{12} - 10 \leq 0 \\
g_3(x) &= 2x_2 + 2X_3 + x_{11} + x_{12} - 10 \leq 0 \\
g_4(x) &= -8x_1 + x_{10} \leq 0 \\
g_5(x) &= -8x_2 + x_{11} \leq 0 \\
g_6(x) &= -8x_3 + x_{12} \leq 0 \\
g_7(x) &= -2x_4 - x_5 + x_{10} \leq 0 \\
g_8(x) &= -2x_6 - x_7 + x_{11} \leq 0 \\
g_9(x) &= -2x_8 - x_9 + x_{12} \leq 0 \\
0 \leq x_d &\leq 1,\ d = 1, \ldots, 9, 13 \\
0 \leq x_d &\leq 100,\ d = 10, 11, 12
\end{aligned}
\tag{3}
$$

$$\begin{aligned} \text{Max } f(x) = & \frac{\sum_{d=1}^{D} \cos^4(x_d) - 2\prod_{d=1}^{D} \cos^2(x_d)}{\sqrt{\sum_{d=1}^{D} dx_d^2}} \\ \text{s.t. } & g_1(x) = 0.75 - \prod_{d=1}^{D} x_d \leq 0 \\ & g_2(x) = \sum_{d=1}^{D} x_d - 7.5D \leq 0 \\ & 1 \leq x_d \leq 10,\ 1, \ldots, 20 \end{aligned} \tag{4}$$

$$\begin{aligned} \text{Min } f(x) = & x_1^2 + (x_2 - 1)^2 \\ \text{s.t. } & g_1(x) = x_2 - x_1^2 = 0 \\ & -1 \leq x_1, x_2 \leq 1 \end{aligned} \tag{5}$$

Para a resolução das equações foram feitos alguns ajustes no algoritmo do PSO-H. Uma das alterações consistiu na aplicação de penalidades ao valor de aptidão de uma partícula caso esta violasse alguma das restrições do problema. A penalidade consiste na atribuição de um valor a aptidão da partícula com o intuito de piorar o fitness. E em seguida novos valores de forma aleatória são atribuídos as dimensões da partícula como tentativa de melhor a aptidão em um futuro próximo. A penalidade aplicada pode ser vista no pseudocódigo a seguir.

```
Penalidade = 1,49
if Partícula atende restrições then
   Retorne Aptidão da partícula
else
   for i = 1 até 10% do numeroDimensõesParticula do
      x = Rand() //[1,D]
      P[x] = Valor //[0,1]
   end for
   Retorne Aptidãopartícula * Penalidade
end if
```

Tabela 1: Parâmetros do PSO-H

Parâmetros	**Valores**
c_1, c_2	1.49445
W	$0.5 + (Rand()/2)$
Iterações	300
Tamanho da Nuvem	30
Taxa de mutação	10%

A Equação 5 possui como restrição uma equação, que fora modificada para tornar-se uma inequação, cujo valor inicial de δ é igual a 0.01 e no decorrer das iterações do algoritmo decresce gradativamente para 0.0001, a alteração ocorreu como mostrado abaixo:

Antes: $g(x) = x_2 - x_1^2 = 0$

Depois: $g(x) = x_2 - x_1^2 \leq \delta$

Tanto o PSO canônico quanto o PSO-H foram implementados em linguagem Java e os testes do algoritmo foram realizados em um computador com as seguintes configurações: processador Intel core 2 duo 2.20 GHz, 3GB de memória, HD 500GB e IDE Eclipse Hélios. O algoritmo foi rodado trinta e uma vezes em cada um dos problemas para permitir uma comparação estatística entre os algoritmos evolutivos considerados. As Tabelas 2 e 3 mostram os resultados obtidos nas funções de benchmarks consideradas no PSO e no PSO-H, respectivamente.

Comparando os resultados apenas através das tabelas, o PSO-H apresenta melhores resultados do que o PSO, especialmente em se tratando do benchmark E_1. Além disso, ambos os algoritmos evolutivos apresentam um bom comportamento tanto em E_2 quanto em E_3, o que pode ser visto através do baixo desvio padrão. Isso significa que na grande maioria das vezes, ambos encontram soluções próximas do ótimo, com exceção do PSO em E_1 que apresentou um desvio relativamente alto.

Tabela 2: Resultado da simulação - PSO

Eq.	Ótimo	Melhor	Médio	Desvio
E_1	-15,0	-11,0306	-7,9990	3,0042
E_2	0.8036	0,7860	0,7258	0,1714
E_3	0,75	0,7709	0,9558	0,3107

Tabela 3: Resultado da simulação PSO-H

Eq.	Ótimo	Melhor	Médio	Desvio
E_1	-15,0	-14,9916	-14,9154	0,0509
E_2	0.8036	0,7960	0,7586	0,1449
E_3	0,75	0,750	0,9545	0,2498

Apesar dos resultados evidenciarem que o PSO-H apresentou melhor desempenho em E_1, é necessário realizar uma avaliação estatística para ratificar as considerações efetuadas. A Tabela 4 apresenta uma comparação estatística, isto é, um *teste-t* com grau de liberdade 30 e nível de significância $\alpha = 0,05$, considerando como hipótese nula que as médias são iguais (h_0). Este nível de significância indica que qualquer valor de t fora do intervalo $-2,6395 < t < 2.6395$ deve rejeitar a hipótese nula, sendo que o valor de t é obtido pela Equação 6, onde S_i representa a média do grupo i, δ_i é o desvio padrão do grupo i e n a quantidade de execuções.

$$t = \frac{S_1 - S_2}{\sqrt{\frac{\delta_1^2}{n} + \frac{\delta_2^2}{n}}} \tag{6}$$

Tabela 4: Comparativo entre PSO e PSO-H

Eq.	Desv PSO	Desv PSO-H	t
E_1	3,0042	0,0509	12,8163
E_2	0,1714	0,1449	-0,8119
E_3	0,3107	0,2498	0,0187

Os valores de t encontrados na Tabela 4 realmente corroboram a afirmação de que em E_1 o PSO-H apresenta melhores resultados, pois 12,8163 esta fora do intervalo definido para $\alpha = 0,05$, rejeitando h_0. Por outro lado, em E_2 e E_3 não se pode rejeitar h_0, significando que estatisticamente, nesses benchmarks, o PSO e o PSO-H apresentaram um desempenho semelhante.

5.1 Comparação com a literatura

Como mencionado, o trabalho de [10] faz uma hibridização de PSO com AG introduzindo no PSO as operações de seleção via roleta, cruzamento e mutação. Nesse contexto, os resultado do PSO-H são comparados com os resultados do trabalho de Yang, que foram chamados de PSO-Y. A Tabela 5 mostra os melhores resultados obtidos por ambos os trabalhos considerando 1000 iterações, já que Yang não apresenta nem a média, nem o desvio padrão.

Tabela 5: Resultado da simulação - PSO-H vs PSO-Y (Melhor)

Eq.	Ótimo	PSO-Y	PSO-H
E_3	-15,0	-14.9990	-14,9916
E_4	0.8036	0.79930	0,7960
E_5	0,75	0,750	0,750

Levando em consideração apenas os melhores resultados pode-se verificar que os algoritmos apresentam resultados bem semelhantes. No entanto considera-se que os resultados do PSO-H são melhores, dado que o poder computacional necessário para adicionar todas as operações consideradas por Yang é bem maior do que o necessário para adicionar apenas a mutação, como foi feito com o PSO-H.

6 Conclusões

A exploração de técnicas de otimização com o intuito de aplicá-las na resolução de problemas com restrições é algo extremamente interessante tanto no campo da informática quanto em outras áreas, como por exemplo, as engenharias. O estudo de algoritmos como o PSO nos mostra que com algumas modificações, adaptações, seu rendimento pode se equiparar ou mesmo superar modelos tradicionais tal como os AGs.

O algoritmo de otimização de partículas híbrido PSO-H mostrado neste trabalho apresentou um bom desempenho, tanto em maximização quanto na minimização das funções consideradas, o que pode significar que o futuro dos algoritmos de otimização pode estar na hibridização, isto é, ma fusão das melhores características dos algoritmos de otimização visando à criação de algoritmos cada vez mais eficientes.

Apesar dos bons resultados apresentados, o algoritmo precisa continuar sendo estudado e ser avaliado com mais funções de referência e diferentes formas de mutação. A utilização de lógica nebulosa para o controle dos parâmetros do PSO-H também pode ser levada em consideração.

REFERÊNCIAS

[1] L. J. Fogel, A. J. Owens and M. J. Walsh. *Artificial Intelligence Through Simulated Evolution*. John Wiley and Sons, New York, NY, 1996.

[2] H. P. Schwefel. *Evolution and Optimum Seeking*. John Wiley and Sons, New York, NY, 1996.

[3] J. Kennedy and R. Eberhurt. "Particle Swarm Optimization". In *IEEE International Conference on Neural Networks*, volume 4, 1997.

[4] R. C. Eberhart and Y. Shi. "Evolving Artificial Neural Networks". In *International Conference on Neural Networks and Brain*, 1998.

[5] R. Thangraj, M. Pant, A. Abraham, K. Deep and V. . Snasel. "Differential Evolution Using a Localized Cauchy Mutation Operator". In *IEEE International Conference on Systems Man and Cybernetics*, pp. 3710 –3716, 2010.

[6] Y. Gao and S. Jia. "Hybrid Differential Evolution Algorithm with Annealing and Chaos". In *International Conference on Natural Computation*, pp. 270–274. IEEE Computer Society, 2009.

[7] A. B. Hashemi and M. R. Meybodi. "Adaptive Parameter Selection Scheme for PSO: A Learning Automata Approach". In *International CSI Computer Conference*. IEEE Press, 2009.

[8] M. Gong, L. Jiao, W. Ma and R. Shang. "Hybrid Immune Algorithm with Intelligent Recombination". In *IEEE Congress on Evolutionary Computation, CEC*, pp. 1807 –1814, 2009.

[9] S. A. Sarah, I. J. Daisy and R. Varghese. "Optimization of Process Parameters Using Genetic Algorithm and PSO". In *International Conference on Communication and Computational Intelligence (INCOCCI)*, pp. 340 –345, dec. 2010.

[10] B. Yang, Y. Chen and Z. Zhao. "A Hybrid Evolutionary Algorithm by Combination of PSO and GA for Unconstrained and Constrained Optimization Problems". In *IEEE International Conference on Control and Automation*. IEEE Computer Society, 2007.

[11] N. Li, Y.-Q. Qin, D.-B. Sun and T. Zou. "Particle swarm optimization with mutation operator". In *Proceedings of 2004 International Conference on Machine Learning and Cybernetics*, volume 4, pp. 2251 – 2256, aug. 2004.

[12] C. Li, S. Yang and I. Korejo. "A Modified Particle Swarm Optimization Algorithm". In *Proceedings of the Fourth International Conference on Machine Learning and Cybernetics*, edited by I. Press, pp. 18–21, 2005.

[13] H. Muhlenbein and D. Schlierkamp-Voosen. "Predictive Models for the breeder genetic algorithm". *Journal Evolutionary Computation*, vol. 1, no. 1, pp. 25–49, 1993.

[14] C. Li, S. Yang and I. Korejo. "An adaptive mutation operator for particle swarm optimization". In *Proc. of the 2008 UK Workshop on Computational Intelligence*, pp. 165–170, 2008.

[15] X. Yao, L. Y. and G. Lin. "Evolutionary programming made faster". *Evolutionary Computation, IEEE Transactions on*, vol. 3, no. 2, pp. 82 –102, jul 1999.