Asociación Argentina
de Mecánica Computacional

Mecánica Computacional Vol XXIX, págs. 9615-9627 (artículo completo)
Eduardo Dvorkin, Marcela Goldschmit, Mario Storti (Eds.)
Buenos Aires, Argentina, 15-18 Noviembre 2010

OTIMIZAÇÃO POR COLÔNIA DE BACTÉRIAS APLICADA A PROBLEMAS DE ENGENHARIA.

Fábio Roberto Teodoro[a], **Rafael Stubs Parpinelli**[a,b] **e Heitor Silvério Lopes**[a]

[a]*Laboratório de Bioinformática, Universidade Tecnológica Federal do Paraná, Av. 7 de setembro 3165, 80230-901, Curitiba, Brasil , E-mails: fr.teodoro@gmail.com, hslopes@utfpr.edu.br*

[b]*Grupo de Computação Cognitiva Aplicada, Universidade do Estado de Santa Catarina, Campus Universitário Prof. Avelino Marcante s/n, 89223-100, Joinville, Brasil , E-mail: parpinelli@joinville.udesc.br*

Palavras Chave: Otimização por Colônia de Bactérias, Algoritmos de Otimização, Engenharia Estrutural.

Resumo. A Otimização por Colônia de Bactérias (*Bacterial Foraging Optimization* – BFO) é uma metaheurística inspirada no comportamento de bactérias Escherichia coli na busca por nutrientes em seu ambiente. Este trabalho propõe uma melhoria ao algoritmo básico pela combinação de duas variações já conhecidas deste: BFO Adaptativo (BFO-A) e BFO com *swarming* (BFO-S). No algoritmo BFO-A, cada bactéria possui o seu tamanho de passo individualizado. Isto influencia o seu comportamento que alterna entre busca global (onde o tamanho do passo é maior) e busca local (onde se reduz o tamanho do passo). Na variante BFO-S, o comportamento de *swarming* é implementado fazendo com que cada bactéria atraia as outras para perto de si, modificando o espaço de busca no seu entorno. Isto reflete a tendência das bactérias reais de permanecerem próximas umas das outras. De fato, elas também repelem outras bactérias muito próximas, numa analogia ao consumo de nutrientes pelas bactérias. Assim se cria uma distância mínima entre as bactérias. O algoritmo proposto neste trabalho, chamado de BFO-SA, combina estas duas estratégias com o intuito de verificar o comportamento do algoritmo. Para análise do desempenho são usadas funções multimodais com e sem restrições e dois problemas de Engenharia Estrutural: (a) minimização do peso de uma treliça plana com dez barras, respeitando restrições de esforço e deslocamento; (b) minimização do custo de fabricação de uma viga soldada respeitando restrições de esforço e peso suportado. São realizadas comparações também entre o algoritmo BFO-SA e os algoritmos BFO-A e BFO-S usando as mesmas funções dos artigos originais. Verificou-se que, em geral, a variante BFO-A apresenta soluções de melhor qualidade que as demais.

1 INTRODUÇÃO

Existem muitas abordagens usando métodos numéricos para resolver problemas de engenharia. Porém, quando o problema apresenta um espaço de busca multimodal ou com muitas restrições, a solução encontrada por um método numérico vai depender muito do ponto no espaço onde se inicia a busca (Lee e Geem, 2005). Por isso, frequentemente meta-heurísticas inspiradas na natureza, como o algoritmo *Bacterial Foraging Optimization* (BFO), se apresentam como alternativas interessantes de solução.

O algoritmo *Bacterial Foraging Optimization* – BFO (Passino, 2002) está inserido dentro da área de Inteligência de Enxames e inspira-se no comportamento de busca por nutrientes da bactéria *Escherichia coli*. Existem diversas variantes do BFO e aqui são abordadas duas delas. A primeira, uma variante com auto-adaptação do comportamento das bactérias (*Autoadaptative BFO* – BFO-A) onde o tamanho dos passos de movimento das bactérias se altera de acordo com as características da região do espaço de busca sendo explorado por aquela bactéria (Chen et al., 2008). A segunda variante tem o comportamento de *swarming* (*Swarming BFO* – BFO-S), onde as bactérias tendem a ficar próximas umas das outras e a se movimentarem em conjunto (Passino, 2002). Neste trabalho propõe-se uma combinação destas duas variantes, denominada (*Swarming and Autoadaptative BFO* – BFO-SA).

Aqui, os algoritmos BFO (original), BFO-A, BFO-S e BFO-SA foram aplicados a diversos problemas de otimização, utilizados extensivamente como *benchmark* para teste de desempenho de metaheurísticas. Foram utilizadas as funções matemáticas multimodais Rastrigin e Griewank, e funções com restrições Himmelblau e *Constrained* II. Além destas, os algoritmos também foram aplicados a dois problemas de Engenharia Estrutural (a) minimização do peso de uma treliça plana com dez barras, respeitando restrições de esforço e deslocamento; (b) minimização do custo de fabricação de uma viga soldada, respeitando restrições de esforço e peso suportado.

Este trabalho está estruturado da seguinte forma: na Seção 2 está descrito o algoritmo BFO juntamente com suas variantes; na Seção 3 estão descritos os problemas solucionados pelo BFO e suas variantes; na Seção 4 é feita a análise dos resultados obtidos; na Seção 5 são apresentadas conclusões e possíveis trabalhos futuros.

2 O ALGORITMO BFO E SUAS VARIANTES

Nesta seção o algoritmo BFO canônico é descrito. Na sequência, são descritas também duas variações deste algoritmo: BFO Adaptativo (BFO-A) proposto por (Chen et al., 2008), e o BFO com *swarming* (BFO-S) apresentado por (Passino, 2002).

O Algoritmo 1 a seguir apresenta o pseudo-código do BFO básico, conforme descrito em (Chen et al., 2008). Nele é gerada uma população de S bactérias no espaço de busca, onde cada bactéria representa uma possível solução para o problema de otimização sendo tratado. São executadas N_c etapas de quimiotaxia (*chemotaxis*) ao fim das quais é executada a etapa de reprodução. Isso é repetido N_{re} vezes e, ao fim, é executada a etapa de dispersão. E tudo isso é repetido N_{ed} vezes.

No movimento de quimiotaxia a bactéria dá um passo em uma direção aleatória. Caso esse passo represente uma melhora na qualidade *fitness* da bactéria, inicia-se então o movimento *swim*, em que a bactéria se move na mesma direção enquanto o *fitness* continuar diminuindo ou até alcançar o número máximo de passos de *swim* (N_s). Isto é feito para cada uma das bactérias. Nesta versão do algoritmo o tamanho do passo é prefixado em $C(i)$.

Na etapa de reprodução as S_r bactérias mais saudáveis são replicadas e introduzidas no

Algoritmo 1 BFO Canônico

Inicialize os parâmetros: $n, S, Nc, Ns, Nre, Ned, Ped, Sr, C^i, \theta^i (i = 1, 2, \cdots, S)$ {θ^i representa a posição da i-ésima bactéria}
$l \leftarrow 0$
repeat
  $k \leftarrow 0$
  repeat
    $j \leftarrow 0$
    repeat {etapa de quimiotaxia}
      for all i **do**
        $J_{last} \leftarrow J(i, j, k, l)$ {*fitness* da i-ésima bactéria}
        Gere um vetor unitário aleatório: Δ^i
        $\theta^i(j + 1, k, l) \leftarrow \theta^i(j, k, l) + \Delta^i C^i$ {nova posição da bactéria}
        Calcule $J(i, j + 1, k, l)$ com $\theta^i(j + 1, k, l)$ {novo *fitness* da bactéria}
        $m \leftarrow 0$
        while $m < Ns$ e $J(i, j + 1, k, l) < J_{last}$ **do** {*swim*}
          $m \leftarrow m + 1$
          Mova a bactéria mais um passo, de comprimento C^i, na direção Δ^i
          Calcule $J(i, j + 1, k, l)$ com o novo $\theta^i(j + 1, k, l)$
        end while
      end for
      $j \leftarrow j + 1$
    until $j \geq Nc$
    for all i **do** {calcula a "saúde" das bactérias}
      $J^i_{health} \leftarrow \sum_{j=0}^{Nc} J(i, j, k, l)$
    end for
    Substitua as Sr bactérias com os maiores J_{health} por cópias das Sr bactérias com os menores J_{health} {reprodução}
    $k \leftarrow k + 1$
  until $k \geq Nre$
  Com probabilidade Ped, atribua uma posição aleatória para cada bactéria {dispersão}
  $l \leftarrow l + 1$
until $l \geq Ned$

mesmo local do espaço busca dos seus "pais", e as S_r bactérias menos saudáveis são eliminadas. A saúde das bactérias é o somatório dos *fitness* para todos os passos de quimiotaxia realizados. Quanto menor o valor, mais saudável é a bactéria e, consequentemente, melhor é a qualidade da solução que ela representa.

Na etapa de dispersão, cada bactéria pode ser eliminada e substituída por outra gerada aleatoriamente, de acordo com a probabilidade P_{ed}.

Por definição, o BFO é um algoritmo de minimização e tem os seguintes parâmetros de controle:

- n: número de dimensões do espaço de busca;
- S: número de bactérias;
- N_c: número de etapas de quimiotaxia;
- N_s: número máximo de passos de *swim*;
- N_{re}: número de etapas de reprodução;
- N_{ed}: número de etapas de dispersão;
- P_{ed}: probabilidade de dispersão;
- $C(i)(i = 1, 2, \ldots, S)$: comprimento do passo nos movimentos da bactéria;
- S_r: número de bactérias replicadas a cada etapa de reprodução.

2.1 BFO Auto-Adaptativo

No BFO-A cada bactéria possui o seu tamanho de passo individualizado. Isto reflete no comportamento da bactéria, alternando entre busca global (onde o tamanho do passo é maior), quando se percebe estagnação da bactéria, e busca local (onde se reduz o tamanho do passo), quando a bactéria encontra uma região promissora do espaço de busca (Chen et al., 2008).

Nesta variante do algoritmo são acrescentados ao BFO básico os parâmetros α, β, ε e Nu. Após a etapa de quimiotaxia de cada bactéria, um contador de estagnação ($no\_improv\_count^i$) é atualizado. Este é incrementado caso a bactéria não melhore o seu *fitness* e zerado caso contrário. O parâmetro α é o fator de redução do tamanho do passo (C^i) das bactérias. O parâmetro β é o fator de redução do parâmetro ε, que é o limiar (considerando o valor do *fitness*) a partir do qual a bactéria tem seu tamanho de passo reduzido. O parâmetro Nu é o número de etapas de quimiotaxia que uma bactéria pode ficar estagnada antes de retornar ao modo de busca global.

Caso haja uma melhoria no *fitness* da bactéria após o passo de quimiotaxia no tempo *t*, seu tamanho de passo no próximo instante de tempo ($C^i(t+1)$) é atualizado conforme a Equação 1, podendo a bactéria entrar em modo de busca local. Também é atualizado o parâmetro ε^i associado a esta bactéria de acordo com a Equação 2.

$$C^i(t+1) = \begin{cases} C^i(t)/\alpha & \text{, se } J(i,j,k,l) < \varepsilon^i(t) \\ C^i(t) & \text{, se } J(i,j,k,l) \geq \varepsilon^i(t) \end{cases} \quad (1)$$

$$\varepsilon^i(t+1) = \begin{cases} \varepsilon^i(t)/\beta & \text{, se } J(i,j,k,l) < \varepsilon^i(t) \\ \varepsilon^i(t) & \text{, se } J(i,j,k,l) \geq \varepsilon^i(t) \end{cases} \quad (2)$$

Caso não haja uma melhoria no *fitness* da bactéria após o passo de quimiotaxia, os parâmetros C^i e ε^i são atualizados conforme as equações 3 e 4, podendo retornar ao estado de busca global.

$$C^i(t+1) = \begin{cases} C^i_{initial} & \text{, se } no\_improv\_count^i > Nu \\ C^i(t) & \text{, se } no\_improv\_count^i \leq Nu \end{cases} \tag{3}$$

$$\varepsilon^i(t+1) = \begin{cases} \varepsilon^i_{initial} & \text{, se } no\_improv\_count^i > Nu \\ \varepsilon^i(t) & \text{, se } no\_improv\_count^i \leq Nu \end{cases} \tag{4}$$

Chen et al. (2008) observaram que a qualidade das soluções encontradas pelo algoritmo BFO é bastante dependente do tamanho do passo das bactérias. Assim, a variante BFO-A foi proposta por estes autores como forma de contornar o problema.

2.2 BFO com *Swarming*

Algumas espécies de bactérias, dentre elas a Escherichia Coli, possuem o comportamento de se moverem em grupo, caracterizando o *swarming*. Este comportamento ocorre pela liberação de feromônios que atraem as bactérias (Passino, 2002).

Segundo Passino (2002), o comportamento de *swarming* é implementado no algoritmo BFO-S fazendo com que cada bactéria atraia as outras para perto de si, modificando o espaço de busca no seu entorno de acordo com a Equação 5. Este comportamento reflete a tendência das bactérias reais de permanecerem próximas umas das outras. Elas também repelem outras bactérias muito próximas, numa analogia ao consumo de nutrientes pelas bactérias, mantendo uma distância mínima entre as elas.

Quatro novos parâmetros de controle são introduzidos no BFO-S (Passino, 2002): o parâmetro $d_{attract}$ representa a profundidade do atrator liberado pelas bactérias, o parâmetro $w_{attract}$ corresponde à largura do sinal de atração, e os parâmetros $h_{repellant}$ e $w_{repellant}$ são a altura do repelente e largura do repelente, respectivamente.

A distorção do espaço de busca onde as bactérias se movimentam é realizada pela função J_{cc} (Equação 5) que é somada à função de *fitness*. O seu efeito é fazer com que as bactérias tendam a permanecer próximas umas das outras. Nesta equação, θ representa uma posição do espaço de busca, $P(j,k,l)$ é o conjunto de posições de todas as bactérias na etapa de quimiotaxia j da etapa de reprodução k da etapa de dispersão l, θ^i é a posição da bactéria i, θ_m representa o componente m do vetor da posição θ e θ^i_m o componente m de θ^i.

$$\begin{aligned} J_{cc}(\theta, P(j,k,l)) &= \sum_{i=1}^{S} J^i_{cc}(\theta, \theta^i(j,k,l)) \\ &= \sum_{i=1}^{S} \left[-d_{attract} \cdot e^{\left(-w_{attract} \sum_{m=1}^{p} (\theta_m - \theta^i_m)^2 \right)} \right] \\ &\quad + \sum_{i=1}^{S} \left[h_{repellant} \cdot e^{\left(-w_{repellant} \sum_{m=1}^{p} (\theta_m - \theta^i_m)^2 \right)} \right] \end{aligned} \tag{5}$$

2.3 BFO Auto-Adaptativo com *Swarming*

No BFO-SA são combinadas as características das duas variantes mencionadas anteriormente, BFO-A e BFO-S. Desta forma, as bactérias devem se movimentar em conjunto, cada

uma influenciando o movimento de suas vizinhas, bem como ajustar o tamanho de seus passos de quimiotaxia. O efeito global é uma exploração mais eficiente das regiões promissoras do espaço de busca. Esta variante utiliza os mesmos parâmetros do BFO-A e BFO-S.

3 EXPERIMENTOS COMPUTACIONAIS

Para analisar a aplicabilidade e eficiência do BFO e suas variantes foram realizados experimentos usando funções multimodais com e sem restrições e também problemas de engenharia, todos problemas de minimização. Cada experimento foi executadas 100 vezes e foi estabelecido um limite de 500.000 avaliações de *fitness* para cada um.

Todos os experimentos foram realizados em computadores com a mesma configuração: computador *desktop* com com processador *core-2 Quad* de 2.8 GHz, 2GB de RAM e rodando sobre uma instalação mínima do sistema operacional Arch Linux. A linguagem de desenvolvimento utilizada foi ANSI C.

3.1 Funções de *Benchmark*

Foram utilizadas as funções matemáticas de Rastrigin e de Griewank, bastante conhecidas e utilizadas como *benchmark* para algoritmos de otimização (Chen et al., 2008). Para cada uma delas foram feitos experimentos com 10, 30 e 50 dimensões.

A função de Rastrigin é uma função não-linear, não-convexa e fortemente multimodal, originalmente proposta para duas dimensões, mas posteriormente generalizada para *n* dimensões (Mühlenbein et al., 1991). A função é definida pela equação 6:

$$f(\vec{x}) = \sum_{i=1}^{n}(x_i^2 - 10cos(2\pi x_i) + 10) \tag{6}$$

Onde $\vec{x}$ é o vetor de parâmetros de dimensão n e os valores de x_i estão no intervalo $[-5.12, 5.12]$. A solução ótima é $(x_1, x_2, \ldots, x_n) = (0, 0, \ldots, 0)$, com o valor de $f(\vec{x}) = 0$.

A função de Griewank é também não-linear multimodal e tem um número de mínimos locais que cresce exponencialmente à medida que aumenta o número de dimensões (Griewank, 1981). A função é definida pela equação 7:

$$f(\vec{x}) = \frac{1}{4000}\left(\sum_{i=1}^{n} x_i^2\right) - \left(\prod_{i=1}^{n} cos\left(\frac{x_i}{\sqrt{i}}\right)\right) + 1 \tag{7}$$

Onde $\vec{x}$ é o vetor de parâmetros de dimensão n e os valores de x_i estão no intervalo $[-600, 600]$. A solução ótima é $(x_1, x_2, \ldots, x_n) = (0, 0, \ldots, 0)$, com o valor de $f(\vec{x}) = 0$.

3.2 Funções com Restrições

Duas funções com restrições foram utilizadas para verificar o desempenho dos algoritmos nestas condições.

A primeira função é conhecida como função não-linear de Himmelblau (Himmelblau, 1972). Ela é definida pela equação 8, contendo cinco variáveis, seis restrições não-lineares, e dez condições de contorno (equações 9–17).

$$f(\vec{x}) = 5.3578547x_3^2 + 0.835689x_1x_5 + 37.293239x_1 - 40792.141 \tag{8}$$

Sujeita a:

$$g_1(\vec{x}) = 85.334407 + 0.0056858x_2x_5 + 0.00026x_1x_4 - 0.0022053x_3x_5 \tag{9}$$

$$g_2(\vec{x}) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 - 0.0021813x_3^2 \quad (10)$$
$$g_3(\vec{x}) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 - 0.0019085x_3x_4 \quad (11)$$
$$0 \leq g_1(\vec{x}) \leq 92 \quad (12)$$
$$90 \leq g_2(\vec{x}) \leq 110 \quad (13)$$
$$20 \leq g_3(\vec{x}) \leq 25 \quad (14)$$
$$78 \leq x_1 \leq 102 \quad (15)$$
$$33 \leq x_2 \leq 45 \quad (16)$$
$$27 \leq x_i \leq 45 \ (i = 3, 4, 5) \quad (17)$$

O melhor resultado conhecido para esta função é: $f(\vec{x}) = -31024.3166$, e foi obtido por (Fesanghary et al., 2008).

A segunda função, identificada como *Constrained* II foi proposta por (Hock e Schittkowski, 1980) e é definida pela equação 18. A função tem sete variáveis e quatro restrições não-lineares (equações 19–23).

$$f(\vec{x}) = (x_1-10)^2+5(x_2-12)^2+x_3^4+3(x_4-11)^2+10x_5^6+7x_6^2+x_7^4-4x_6x_7-10x_6-8x_7 \quad (18)$$

Sujeita a:

$$g_1(\vec{x}) = 127 - 2x_1^2 - 3x_2^4 - x_3 - 4x_4^2 - 5x_5 \geq 0 \quad (19)$$
$$g_2(\vec{x}) = 282 - 7x_1 - 3x_2 - 10x_3^2 - x_4 + x_5 \geq 0 \quad (20)$$
$$g_3(\vec{x}) = 196 - 23x_1 - x_2^2 - 6x_6^2 + 8x_7 \geq 0 \quad (21)$$
$$g_4(\vec{x}) = -4x_1^2 - x_2^2 + 3x_1x_2 - 2x_3^2 - 5x_6 + 11x_7 \geq 0 \quad (22)$$
$$-10 \leq x_i \leq 10 \ (i = 1, \ldots, 7) \quad (23)$$

Para esta função de teste, o melhor resultado conhecido é: $f(\vec{x}) = 680.6300577$, e foi obtido por (Fesanghary et al., 2008).

3.3 Problemas de Engenharia Estrutural

Dois problemas de Engenharia Estrutural foram abordados: cálculo de uma treliça plana com 10 barras e dimensionamento de uma viga soldada.

3.3.1 Treliça Plana de 10 Barras

Neste problema o objetivo é minimizar o peso total de uma treliça plana de 10 barras. O peso desta treliça é dado pela equação 24:

$$f(\vec{x}) = \sum_{j=1}^{10} \rho A_j L_j \quad (24)$$

onde A_j é a área da seção transversal da j-ésima barra, L_j o comprimento dessa barra e ρ é a densidade do material.

Este problema já foi extensivamente utilizado na literatura para teste de métodos de otimização, como por exemplo: (Lee e Geem, 2005), (Schmit e Miura, 1976), (Venkayya, 1971), (Rizzi, 1976), (Khan et al., 1979) e (Fesanghary et al., 2008). Neste trabalho foram usadas as mesmas constantes e restrições usadas nos trabalhos citados. Para este problema o melhor resultado conhecido é: $f(\vec{x}) = 4668.72$ e foi obtido por (Fesanghary et al., 2008).

3.3.2 Viga Soldada

Neste problema o objetivo é minimizar o custo de fabricação de uma viga soldada. Este custo é definido pela equação 25, onde $h = x_1$ representa a espessura do cordão de solda, $l = x_2$ o comprimento do cordão de solda, $t = x_3$ a altura da viga e $b = x_4$ a largura da viga (Deb e Srinivasan, 2006).

$$f(\vec{x}) = 1.10471h^2l + 0.04811tb(14.0 + l) \tag{25}$$

Sujeito às seguintes restrições:

$$g_1(\vec{x}) = 13600 - \tau(\vec{x}) \geq 0 \tag{26}$$
$$g_2(\vec{x}) = 30000 - \sigma(\vec{x}) \geq \tag{27}$$
$$g_3(\vec{x}) = b - h \geq 0 \tag{28}$$
$$g_4(\vec{x}) = Pc(\vec{x}) - 6000 \geq 0 \tag{29}$$
$$0.125 \leq h, b \leq 5 \tag{30}$$
$$0.1 \leq l, t \leq 10.0 \tag{31}$$

onde: $\tau(\vec{x})$, $\sigma(\vec{x})$ e $Pc(\vec{x})$ representam, respectivamente, a tensão de cisalhamento, a tensão de flexão e a carga de dobra. Os detalhes de cálculo destes parâmetros podem ser encontrado na literatura.

Este problema foi também abordado por (Coello, 2000b), (Coello, 2000a), (Lee e Geem, 2005), (Ragsdell e Phillips, 1976), (Deb, 1991), (Deb, 2000), (Mahdavi et al., 2007) e (Fesanghary et al., 2008). O melhor resultado conhecido é $f(\vec{x}) = 1.7248$ e foi obtido por (Mahdavi et al., 2007) e também por (Fesanghary et al., 2008).

4 RESULTADOS

Os resultados obtidos com a aplicação dos algoritmos são apresentados sob forma de tabelas, onde 10, 30 e 50 representam o número de dimensões, $fitness_{med}$ e $fitness_{dp}$ são a média e o desvio-padrão, respectivamente, dos resultados obtidos em 100 execuções independentes, e $tempo_{med}$ e $tempo_{dp}$ são a média e desvio-padrão dos tempos de execução dos algoritmos (também para 100 execuções). Em todas as tabelas os dados estão agrupados por algoritmo: BFO, BFO-A, BFO-S e BFO-SA.

Os resultados para a função Rastrigin são apresentados na Tabela 1. Todos algoritmos obtiveram resultados semelhantes, e distantes do ótimo global ($f(\vec{x}) = 0$). Os tempos de execução dos algoritmos BFO e BFO-A foram significativamente melhores do que os BFO-S e BFO-SA.

		10	30	50
BFO	$fitness_{med}$	9,67147	134,76209	295,95523
	$fitness_{dp}$	1,30336	12,14693	25,80825
	$tempo_{med}$	0,45410	1,27410	1,93480
	$tempo_{dp}$	0,00861	0,01619	0,04058
BFO-A	$fitness_{med}$	9,29234	134,25068	297,85532
	$fitness_{dp}$	1,50256	11,42694	23,89999
	$tempo_{med}$	0,43210	1,26690	1,94330
	$tempo_{dp}$	0,00653	0,02352	0,07002
BFO-S	$fitness_{med}$	9,7135	133,64866	298,30019
	$fitness_{dp}$	1,65350	10,20750	23,00123
	$tempo_{med}$	3,68960	11,08910	13,95840
	$tempo_{dp}$	0,05541	0,31959	0,12867
BFO-SA	$fitness_{med}$	9.57853	136.27577	299.52995
	$fitness_{dp}$	1.63396	11.29196	23.07196
	$tempo_{med}$	3.69030	11.04500	13.92850
	$tempo_{dp}$	0.03817	0.30831	0.09254

Tabela 1: Resultados para a função Rastrigin com 10, 30 e 50 dimensões.

Os resultados para a função Griewank são apresentados na Tabela 2. Todos os algoritmos obtiveram resultados semelhantes, porém os tempos de execução dos algoritmos BFO e BFO-A foram significativamente melhores do que os BFO-S e BFO-SA.

		10	30	50
BFO	$fitness_{med}$	0,24617	1,98385	3,49722
	$fitness_{dp}$	0,04662	0,10854	0,23191
	$tempo_{med}$	0,48890	1,53940	2,51770
	$tempo_{dp}$	0,00615	0,00719	0,01737
BFO-A	$fitness_{med}$	0,24441	1,95668	3,52843
	$fitness_{dp}$	0,04394	0,1145	0,23961
	$tempo_{med}$	0,47990	1,53550	2,51120
	$tempo_{dp}$	0,00300	0,01367	0,01211
BFO-S	$fitness_{med}$	0,25336	1,99516	3,54094
	$fitness_{dp}$	0,04490	0,129260	0,20320
	$tempo_{med}$	3,15640	9,08840	12,98460
	$tempo_{dp}$	0,08081	0,14565	0,14741
BFO-SA	$fitness_{med}$	0,24409	1,97844	3,55069
	$fitness_{dp}$	0,04662	0,10577	0,25243
	$tempo_{med}$	3,14370	9,03970	12,91860
	$tempo_{dp}$	0,06577	0,13795	0,07150

Tabela 2: Resultados para a função Griewank com 10, 30 e 50 dimensões.

Para a função Himmelblau, os resultados são mostrados na Tabela 3. A média de resultados para o BFO-A, que foi a variante de BFO que apresentou os melhores resultados, está $\approx 0.03\%$ distante do melhor resultado conhecido. Os tempos de execução dos BFO e BFO-A foram equivalentes e significativamente melhores que os dos BFO-S e BFO-SA.

BFO	$fitness_{med}$	-31013,88758
	$fitness_{dp}$	0
	$tempo_{med}$	0,05190
	$tempo_{dp}$	0,00440
BFO-A	$fitness_{med}$	-31014,01957
	$fitness_{dp}$	0
	$tempo_{med}$	0,05480
	$tempo_{dp}$	0,00500
BFO-S	$fitness_{med}$	-31012,67436
	$fitness_{dp}$	0
	$tempo_{med}$	5,74480
	$tempo_{dp}$	0,14336
BFO-SA	$fitness_{med}$	-31012,09865
	$fitness_{dp}$	0
	$tempo_{med}$	5,75170
	$tempo_{dp}$	0,10713

Tabela 3: Resultados para a função Himmelblau.

Para a função *Constrained* II, os resultados são mostrados na Tabela 4. A média de resultados para o BFO-A, que foi a variante de BFO que apresentou os melhores resultados, está $\approx 0.02\%$ distante do melhor resultado conhecido. Os tempos de execução dos BFO e BFO-A foram equivalentes e significativamente melhores que os dos BFO-S e BFO-SA.

BFO	$fitness_{med}$	680,79457
	$fitness_{dp}$	0,07790
	$tempo_{med}$	0,35670
	$tempo_{dp}$	0,00813
BFO-A	$fitness_{med}$	680,78472
	$fitness_{dp}$	0,05201
	$tempo_{med}$	0,35810
	$tempo_{dp}$	0,01017
BFO-S	$fitness_{med}$	680,80734
	$fitness_{dp}$	0,07571
	$tempo_{med}$	3,30430
	$tempo_{dp}$	0,08405
BFO-SA	$fitness_{med}$	680,81236
	$fitness_{dp}$	0,06893
	$tempo_{med}$	3,29880
	$tempo_{dp}$	0,06341

Tabela 4: Resultados para a função *Constrained* II.

Para o problema da treliça plana com 10 barras, os resultados estão na Tabela 5. A média de resultados para o BFO está $\approx 0.63\%$ distante do melhor resultado encontrado na literatura. Os tempos de execução dos BFO e BFO-A foram equivalentes e significativamente melhores que os dos BFO-S e BFO-SA.

BFO	$fitness_{med}$	4698,08947
	$fitness_{dp}$	5,62599
	$tempo_{med}$	2,72450
	$tempo_{dp}$	0,00921
BFO-A	$fitness_{med}$	4698,24428
	$fitness_{dp}$	5,34640
	$tempo_{med}$	2,71990
	$tempo_{dp}$	0,00592
BFO-S	$fitness_{med}$	4699,96137
	$fitness_{dp}$	5,9008
	$tempo_{med}$	5,93710
	$tempo_{dp}$	0,06643
BFO-SA	$fitness_{med}$	4700,09941
	$fitness_{dp}$	5,79535
	$tempo_{med}$	5,93040
	$tempo_{dp}$	0,03924

Tabela 5: Resultados para o problema da Treliça plana de 10 vigas.

Os resultados obtidos para o problema da viga soldada estão mostrados na Tabela 6. A média de resultados para o BFO-A está $\approx 5.28\%$ distante do melhor resultado encontrado na literatura. Os tempos de execução dos BFO e BFO-A foram equivalentes e significativamente melhores que os do BFO-S e BFO-SA.

BFO	$fitness_{med}$	1,81954
	$fitness_{dp}$	0,04699
	$tempo_{med}$	0,10020
	$tempo_{dp}$	0,00140
BFO-A	$fitness_{med}$	1,81588
	$fitness_{dp}$	0,04162
	$tempo_{med}$	0,10010
	$tempo_{dp}$	0,00099
BFO-S	$fitness_{med}$	1,82895
	$fitness_{dp}$	0,05013
	$tempo_{med}$	13,17490
	$tempo_{dp}$	0,42320
BFO-SA	$fitness_{med}$	1,82448
	$fitness_{dp}$	0,04587
	$tempo_{med}$	13,18360
	$tempo_{dp}$	0,49449

Tabela 6: Resultados para o problema da viga soldada.

De maneira geral, os últimos quatro resultados foram muito bons, considerando que foi usado um algoritmo estocástico e que a média dos resultados obtidos ficou muito próxima da melhor solução conhecida.

5 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho foi comparado o desempenho dos algoritmos BFO, BFO-A, BFO-S e BFO-SA na resolução de problemas de minimização das funções matemáticas sem restrições (Rastrigin e Griewank) com 10, 30 e 50 dimensões, funções com restrições (Himmelblau e *Constrained* II) e problemas de Engenharia Estrutural (Treliça plana de 10 barras e Viga soldada). Os resultados das variantes do BFO foram comparados entre si e a média de resultados da variante com os melhores resultados foi comparada com o melhor resultado encontrado na literatura. Além da qualidade dos resultados foram analisados também os tempos de execução das variantes do BFO. Verificou-se que, em geral, os resultados obtidos pelo BFO-A são um pouco melhores do que os obtidos pelos demais e que o tempo de processamento deste é significativamente menor que o dos BFO-S e BFO-SA, e praticamente igual ao do BFO.

Os resultados obtidos para as funções matemáticas confirmam a afirmação, feita em (Chen et al., 2008), de que o algoritmo *Bacterial Foraging Optimization* (BFO) apresenta um desempenho ruim para problemas com elevada dimensionalidade. Por outro lado, a estratégia de autoadaptação proposta em (Chen et al., 2008) teve desempenho semelhante ao BFO básico nos problemas com grande quantidade de dimensões, apresentando melhora mais significativa em problemas com poucas dimensões.

A estratégia de *swarming* combinada com a de autoadaptação (BFO-SA) apresentou, de maneira geral, desempenho equivalente ao BFO básico. Porém, com tempo de processamento muito maior, o que a torna inviável, pelo menos para os problemas aqui abordados.

Para os problemas de engenharia o BFO-A obteve resultados próximos dos encontrados na literatura, porém utilizando número maior de avaliações.

Uma possível melhoria nos resultados do BFO pode ser obtida pela hibridização com outros algoritmos de otimização. Alguns trabalhos nesse sentido já foram reportados na literatura, como por exemplo (Biswas et al., 2007) onde o BFO foi hibridizado com Otimização por Enxame de Partículas (PSO), conseguindo melhores resultados para funções de grande dimensionalidade. Entretanto, há que se ressaltar que no presente trabalho o objetivo era comparar o desempenho de diferentes variantes do BFO, em especial a BFO-SA. Esta comparação serve de subsídio a desenvolvimentos futuros, híbridos ou não.

A independência entre as bactérias nos seus passos de quimiotaxia, para os algoritmos BFO e BFO-A, leva a crer que uma implementação paralela pode ser bem sucedida, o que possibilitaria tratar, em tempo viável, problemas de maior complexidade ou utilizar populações maiores. Esta será uma das linhas de pesquisa futuras.

REFERÊNCIAS

Biswas A., Dasgupta S., Das S., e Abraham A. Synergy of PSO and bacterial foraging optimization - a comparative study on numerical benchmarks. In E. Corchado, J.M. Corchado, e A. Abraham, editores, *Innovations in Hybrid Intelligent Systems*, volume 44 de *Advances in Soft Computing*, páginas 255–263. Springer, Heidelberg, 2007.

Chen H., Zhu Y., e Hu K. Self-adaptation in bacterial foraging optimization algorithm. In *Proc. 3rd International Conference on Intelligent System and Knowledge Engineering*, páginas 1026–1031. 2008.

Coello C.A.C. Constraint handling through a multiobjective optimization technique. In K. Deb, editor, *Multi-criterion Optimization Using Evolutionary Methods*, páginas 117–118. Orlando, Florida, USA, 2000a.

Coello C.A.C. Use of a self-adaptive penalty approach for engineering optimization problems. *Computers in Industry*, 41(2):113–127, 2000b.

Deb K. Optimal design of a welded beam via genetic algorithms. *AIAA Journal*, 29(11):2013–2015, 1991.

Deb K. An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(2-4):311–338, 2000.

Deb K. e Srinivasan A. Monotonicity analysis, evolutionary multi-objective optimization, and discovery of design principles. Technical Report KanGAL No. 2006004, Indian Institute of Technology, Kanpur Genetic Algorithms Laboratory, 2006.

Fesanghary M., Mahdavi M., Minary-Jolandan M., e Alizadeh Y. Hybridizing harmony search algorithm with sequential quadratic programming for engineering optimization problems. *Computer Methods in Applied Mechanics and Engineering*, 197(33-40):3080–3091, 2008.

Griewank A. Generalized descent for global optimization. *Journal of Optimization Theory and Applications*, 34(1):11–39, 1981.

Himmelblau D.M. *Applied Nonlinear Programming*. McGraw-Hill, New York, USA, 1972.

Hock W. e Schittkowski K. Test examples for nonlinear programming codes. *Journal of Optimization Theory and Applications*, 30, 1980.

Khan M.R., Willmert K.D., e Thornton W.A. An optimality criterion method for large-scale structures. *AIAA Journal*, (17):753–761, 1979.

Lee K. e Geem Z. A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. *Computer Methods in Applied Mechanics and Engineering*, 194(36-38):3902–3933, 2005.

Mahdavi M., Fesanghary M., e Damangir E. An improved harmony search algorithm for solving optimization problems. *Applied Mathematics and Computation*, 188(2), 2007.

Mühlenbein H., Schomisch D., e Born J. The parallel genetic algorithm as function optimizer. *Parallel Computing*, 17(6-7):619–632, 1991.

Passino K. Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Systems Magazine*, 22(3):52–67, 2002.

Ragsdell K. e Phillips D. Optimal design of a class of welded structures using geometric programming. *ASME Journal of Engineering for Industries*, 98(3):1021–1025, 1976.

Rizzi P. Optimization of multi-constrained structures based on optimality criteria. In *Proc. AIAA/ASME/SAE 17th Structures, Structural Dynamics and Materials Conference*, páginas 448–462. 1976.

Schmit L.A. e Miura H. Approximation concepts for efficient structural synthesis. *AIAA Journal*, (12):692–699, 1976.

Venkayya V.B. Design of optimum structures. *Computers & Structures*, (1):1–2, 1971.