Análise de Sensibilidade dos Parâmetros do *Bat Algorithm* e Comparação de Desempenho

Jelson A. Cordeiro[1], **Rafael Stubs Parpinelli**[1] **e Heitor Silvério Lopes**[1]

[1]Laboratório de Bioinformática – Universidade Tecnológica Federal do Paraná (UTFPR)
Curitiba – PR – Brasil

jelsoncordeiro@gmail.com, parpinelli@joinville.udesc.br e hslopes@utfpr.edu.br

Abstract. *This article analyzes the performance of a recently proposed bioinspired algorithm, the Bat Algorithm (BA). First, it is performed a sensitivity analysis of the control parameters of the algorithms. Next, using the best performing set of parameters, comparisons are done with another popular evolutionary computation algorithms, namely, Particle Swarm Optimization (PSO), Differential Evolution (DE) and Artificial Bee Colony (ABC). To evaluate the performance, the quality of solutions was used, submitting the algorithm to the optimization of mathematical benchmark functions. Results show that BA is very promising.*

Resumo. *Este artigo analisa o desempenho de um algoritmo bioinspirado recentemente proposto, o Bat Algorithm (BA). Primeiramente é feita uma análise de sensibilidade dos parâmetros de controle do algoritmo. Posteriormente, utilizando o conjunto de parâmetros que obteve o melhor resultado, são feitas comparações de desempenho com outros algoritmos populares de computação evolucionária: Otimização por Enxame de Partículas (PSO), Evolução Diferencial (DE) e Colônia de Abelhas Artificiais (ABC). Para avaliar o desempenho foi utilizado como critério a qualidade das soluções obtidas na otimização de funções matemáticas de* benchmark. *Os resultados estatísticos mostram que o BA é bastante promissor.*

1. Introdução

A natureza, em especial certos comportamentos animais, tem sido uma fonte inesgotável de inspiração para engenheiros e cientistas da computação. Muitos algoritmos bioinspirados são utilizados para a busca de soluções para os mais diversos problemas do mundo real [Parpinelli e Lopes 2011]. Um dos mais recentes, o Bat Algorithm (BA) [Yang 2010b], foi inspirado na eco-localização desempenhada pelos morcegos durante seu vôo.

O BA é um método bastante recente em comparação com outros métodos de otimização da área de computação evolucionária. Desta forma, a literatura é muito escassa a respeito do método, existindo algumas variantes do mesmo, mas nenhuma definição de parâmetros de controle padrão.

Neste artigo foi feita uma definição formal do BA e realizada uma análise de sensibilidade dos parâmetros do algoritmo. Para tanto, foram utilizadas quatro funções matemáticas frequentemente utilizadas na literatura como *benchmark*: Rastrigin, Griewank, Sphere e Ackley. O objetivo deste estudo é determinar parâmetros padrão para

o algoritmo, estabelecendo um equilíbrio adequado entre exploração e intensificação no processo de otimização. Utilizando os parâmetros padrão, com os quais espera-se obter as melhores soluções possíveis para os problemas tratados, foi feita uma comparação de desempenho com outros métodos populares da área de computação evolucionária: Otimização por Enxame de Partículas (*Particle Swarm Optimization* – PSO) [Kennedy e Eberhart 1995], Evolução Diferencial (*Differential Evolution* – DE) [Storn e Price 1997] e Colônia de Abelhas Artificiais (*Artificial Bee Colony* – ABC) [Karaboga e Akay 2009].

O artigo está organizado da seguinte forma: a seção 2 formaliza o algoritmo BA, e a seção 3 apresenta uma breve revisão dos demais algoritmos que foram utilizados para comparação. Os experimentos e resultados estão, respectivamente, nas seções 4 e 5. Finalmente, conclusões e trabalhos futuros são apresentados na seção 6.

2. BA

O BA [Yang 2010b] é inspirado no processo de eco-localização desempenhada pelos morcegos durante o seu vôo. Os morcegos utilizam esta função tanto para detectar presas, quanto para evitar obstáculos. A eco-localização se baseia na emissão de ondas ultrassônicas e a correspondente medição do tempo gasto para estas ondas voltarem à fonte, após serem refletidas no alvo (presa ou obstáculo). A taxa de pulso e amplitude dos sons emitidos pelos morcegos variam com a estratégia de caça. Quando identificada uma presa, a taxa de pulso (r) é acelerada e a amplitude (A) é aumentada para evitar a perda da presa. Por outro lado, quando a presa está sob domínio, a amplitude diminui.

No modelo computacional correspondente, o BA, cada morcego representa uma possível solução para o problema, codificado sob a forma de um vetor. Uma população de morcegos então se move no espaço de busca do problema, continuamente atualizando a frequência, velocidade e posição de cada elemento buscando encontrar a solução ótima. A cada nova interação, cada morcego é atualizado seguindo a melhor solução encontrada pela população. Além da atualização da posição, existe o controle de exploração e intensificação, como nos outros algoritmos de computação evolucionária. Isto é realizado, respectivamente, pela variação da amplitude e da taxa de pulso.

É importante ressaltar que o autor original do algoritmo não disponibiliza um pseudocódigo do BA [Yang 2010b, Yang 2010a], sendo muitos detalhes omitidos. Assim, propõe-se a formalização do BA no pseudocódigo mostrado no Algoritmo 1, cuja explicação detalhada encontra-se na sequência.

Primeiramente, no instante $t = 0$, todos os n morcegos $\vec{x}_i$ ($i = 1, \ldots, n$) são inicializados com: taxa de pulso $r_i = 0$, velocidade $\vec{v}_i = 0$, amplitude $A_i = 1$, frequência $f_i = 0$ e posição $\vec{x}_i$ aleatoria (linha 2). O ciclo principal representa a evolução da população no tempo (linhas 5-21). O primeiro passo no interior do ciclo é atualizar a posição temporária $\vec{x}_{temp}$ até ser aceita. Para isto, a frequência f_i é atualizada (linha 7), onde f_{min} e f_{max} são os limites inferiores e superiores respectivamente da função de avaliação, e β é um número aleatório. A nova frequência f_i é utilizada para determinar a nova velocidade $\vec{v}_i^{t+1}$ (linha 8), onde $\vec{x}_*$ é a melhor solução encontrada até o instante t. Com a nova velocidade $\vec{v}_i^{t+1}$, é possível determinar a nova posição temporária $\vec{x}_{temp}$ (linha 9).

Na (linha 10) é realizada a busca local, que pode ser implementada de diversas ma-

Algoritmo 1 Bat algorithm (BA)

1: Parâmetros: n, α, λ
2: Inicializa morcegos $\vec{x}_i$
3: Avalia $f(\vec{x}_i)$ para todos os morcegos
4: Atualiza melhor morcego $\vec{x}_*$
5: **while** critério de parada não atingido **do**
6: **for** $i = 1$ **to** n **do**
7: $f_i = f_{min} + (f_{max} - f_{min})\beta, \beta \in [0, 1]$
8: $\vec{v_i}^{t+1} = \vec{v_i}^{t} + (\vec{x_i}^{t} - \vec{x_*}^{t}) f_i$
9: $\vec{x}_{temp} = \vec{x_i}^{t} + \vec{v_i}^{t+1}$
10: **if** $rand < r_i$ **then** {Faz busca local}
11: $\vec{x}_{temp} = \vec{x}_* + \epsilon A_m, \epsilon \in [-1, 1]$
12: **end if**
13: Realiza pertubação em uma dimensão de $\vec{x}_{temp}$
14: **if** $rand < A_i$ **or**$|\ f(\vec{x}_{temp}) \leq f(\vec{x}_i)$ **then** {Aceita solução temporária}
15: $\vec{x}_i = \vec{x}_{temp}$
16: ${r_i}^{t+1} = 1 - exp(-\lambda t)$
17: ${A_i}^{t+1} = \alpha {A_i}^{t}$
18: **end if**
19: Atualiza melhor morcego $\vec{x}_*$
20: **end for**
21: **end while**
22: Pós-processamento

neiras: um passeio aleatório (*random walk*) pode ser usado tanto para exploração quanto intensificação, dependendo do tamanho do passo. Outra maneira é utilizar o operador de mutação não uniforme. Nesta implementação foi utilizado um terceiro método, que foi sugerido pelo criador do BA [Yang 2010b], conforme a equação da linha (11), onde, ϵ é número aleatório, e A_m é a média da amplitude de todos os morcegos em um dado instante t. Na (linha 13) uma dimensão de $\vec{x}_{temp}$, escolhida aleatoriamente entre d dimensões, é modificada aleatoriamente dentro dos limites da função de avaliação.

Se a condição na (linha 14) for verdadeira, a solução temporária $\vec{x}_{temp}$ é aceita (linha 15) e também ocorre o aumento da taxa de pulso (linha 16), sendo que $t \to \infty$ e $r_i \to 1$, a busca local se intensifica com o passar do tempo. Outro valor atualizado é a amplitude A (linha 17). Para controlar a diminuição gradual de A dois métodos (linear e geométrico) foram propostos [Yang 2010a]. Para o método linear, a equação é $A = A_0 - \beta t$, onde A_0 é a amplitude inicial, t é o número da interação e β é taxa de diminuição, tal que $\beta = (A_0 - A_f)/t_f$, sendo A_f a amplitude final e t_f o número máximo de interações. Assim, $A \to 0$ quando $t \to \infty$. Para o método geométrico A diminui com uma taxa de diminuição $0 < \alpha < 1$, utilizando a equação $A = A_0 \alpha^t$, $t = 1, 2, ..., t_f$.

Nesta formalização do BA foi utilizado o segundo método para a diminuição gradual de A, pois tem a vantagem de não precisar especificar o número máximo de interações. É necessário apenas determinar o valor de α e o valor inicial de A. Foi observado que, se a diminuição for lenta o suficiente, o valor encontrado ao final da execução do algoritmo tende a se aproximar do ótimo global [Yang 2010a].

O ciclo principal continua até que a sucessão evolutiva da população de morcegos atinja o critério de parada estabelecido (linha 5), geralmente um número máximo de iterações. Os valores de α e λ são os parâmetros do BA que serão analisados na seção 5.

3. Algoritmos de Computação Evolucionária Baseados em População

Nesta seção será realizada uma breve revisão de alguns algoritmos populares de Computação Evolucionária que serão comparados posteriormente com o BA.

3.1. Algoritmo PSO

O PSO emergiu de experiências que modelam o comportamento social observado em muitas espécies de pássaros e cardumes de peixes, e até mesmo do comportamento social humano. No PSO [Kennedy e Eberhart 1995], os indivíduos da população são representados por pontos, denominados de partículas, que voam no espaço de busca do problema, sendo influenciadas mutuamente. O PSO possui dois tipos de informação importante no processo de decisão: O g_{best} que tente a movimentar a partícula na direção da melhor posição encontrada globalmente pelo enxame, e o p_{best} que tende a movimentar a partícula para a melhor posição encontrada localmente pela própria partícula. Ambos g_{best} e p_{best} são medidas por uma função de avaliação que quantifica a qualidade da solução do problema. Cada partícula se move em uma determinada direção x_i^{t+1}, com velocidade v_i^{t+1}, dada pela Equação 1.

$$v_i^{t+1} = v_i^t + \varphi_1(p_{best} - x_i^t) + \varphi_2(g_{best} - x_i^t) \tag{1}$$

onde φ_1 e φ_2 são os componentes cognitivos e social. Uma vez que a velocidade da partícula é calculada, a nova posição é definida pela Equação 2.

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{2}$$

3.2. Algoritmo ABC

O ABC é inspirado no comportamento de coleta de alimentos pelas abelhas [Karaboga e Akay 2009]. Uma colônia de abelhas consiste em três grupos de abelhas: operárias, seguidores e exploradoras. As abelhas operárias estão associadas com uma fonte de alimento a qual estão explorando. A abelha seguidora espera a dança da operária para tomar a decisão de qual fonte de alimento escolher para explorar. A primeira metade da colônia consiste de abelhas operárias e a segunda de abelhas seguidoras. Para cada fonte de alimento, existe uma abelha operária. Quando a fonte de alimento se extingue, a abelha operária se torna uma abelha exploradora, cujo papel é explorar o ambiente para descobrir novas fontes de alimento. As abelhas exploradoras são caracterizadas por baixo custo de busca e baixa qualidade média da fonte de alimento, mas ocasionalmente, podem descobrir fontes ricas de alimento até então desconhecidas.

Uma colônia de abelhas deve coletar alimentos para sobreviver. Inicialmente, as abelhas do tipo exploradoras são enviadas simultaneamente em múltiplas direções a partir da colméia, de modo a explorar uma grande área de busca. Ao serem encontradas fontes promissoras de alimentos, a colônia de abelhas concentra, então, esforços nestas regiões. As abelhas operárias, através da dança, informam a distância e a localização

exata da fonte de alimento para as demais abelhas na colméia. Assim, outras abelhas são induzidas a buscar alimento nas regiões promissoras, em função da intensidade da dança. Após coletar o alimento e deixá-lo na colméia, uma abelha operária pode: (a) abandonar a fonte de alimento e tornar-se novamente uma abelha exploradora, (b) continuar a coleta, (c) dançar para recrutar mais abelhas antes de retornar para fonte de alimento. As abelhas optam por uma destas alternativas com certa probabilidade.

3.3. Algoritmo DE

A DE [Storn e Price 1997] é um algoritmo baseado em população que utiliza os operadores de cruzamento, mutação e seleção, semelhantes ao algoritmo genético (AG) [Holland 1975]. A principal diferença na construção de melhores soluções é que o AG depende do cruzamento e o DE baseia-se na operação de mutação. Esta operação é baseada nas diferenças de pares de vetores selecionados aleatoriamente na população. A DE utiliza operação de mutação como um mecanismo de busca e o operador de seleção para direcionar a busca para regiões potenciais no espaço de busca.

4. Experimentos

Dois grupos de experimentos foram realizados: ajuste dos parâmetros do BA e comparação do BA com outros algoritmos. Todos os experimentos reportados neste artigo foram realizados usando o mesmo computador com processador i5 de quatro núcleos, com 4GB de RAM, sobre uma instalação mínima do Linux Mint. Os algoritmos foram implementados usando a linguagem C-ANSI.

Cada experimento reportado nas seções seguintes foi executado 20 vezes com sementes aleatórias diferentes. Para diminuir problemas de imprecisão devido a arredondamentos, todos os valores abaixo de 10^{-12} são considerados 0. O critério de parada para cada execução foi estabelecido como o alcance de 500000 avaliações. O tamanho da população do BA foi fixado em $n = 40$, como sugerido por [Yang 2010b].

Para realizar o ajuste dos parâmetros foram utilizadas as funções de *benchmark* mostradas na Tabela 1 para 100 e 200 dimensões.

Tabela 1. Funções matemáticas de *benchmark* utilizadas no ajuste dos parâmetros.

Função	Limites	Dimensões	Equação
Rastrigin	[-5,12...5,12]	100 e 200	$f(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$
Griewank	[-600...600]	100 e 200	$f(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1$
Sphere	[-100...100]	100 e 200	$f(x) = \sum_{i=1}^{n} x_i^2$
Ackley	[-32...32]	100 e 200	$f(x) = -20e^{\left(-0{,}2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right)} - \ldots$ $\ldots - e^{\left(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)\right)} + 20 + e$

O BA foi comparado com o PSO, DE e ABC. O tamanho da população em todos os algoritmos foi fixado com ($n = 40$). Uma outra versão do BA disponível na literatura também foi comparada, uma versão simplificada (BAS) com fonte disponibilizado pelo

criador [Yang 2010b] onde a amplitude e a taxa de pulso são estáticas. A seguir são especificados os parâmetros de controle de cada algoritmo:

PSO : Os componentes cognitivos e social são constantes que podem ser utilizadas para alterar a ponderação entre experiência pessoal e da população. Ambos foram fixados em 1,8 como recomendado por [Vesterstrom e Thomsen 2004].

DE : A constante real f que afeta a variação entre duas soluções foi fixada com 0,5 e o valor da taxa de *crossover*, que controla a diversidade da população, foi fixada com 0,9, como recomendado por [Pham e Karaboga 1991].

ABC : O número limite de interações sem melhoria na solução para ser substituído foi fixado com 100, conforme [Karaboga e Akay 2009].

BAS : A taxa de pulso para controlar a busca local foi fixada em 0,5 e a amplitude para controlar o aceite de soluções ruins e manter a diversidade foi fixada em 0,25, valores definidos por [Yang 2010b].

5. Resultados e Discussão

5.1. Ajuste de Parâmetros do BA

As Tabelas 2 e 3 mostram os resultados dos experimentos para as quatro funções matemáticas com 100 e 200 dimensões, respectivamente. Nestas tabelas, os valores em negrito representam o melhor valor médio obtido para cada função. Neste experimento foi investigado o comportamento dos dois parâmetros principais do BA: α e λ. O parâmetro α que controla a amplitude e, consequentemente, controla a probabilidade de uma solução ruim ser aceita; o parâmetro λ que controla a taxa de pulso e, consequentemente, a probabilidade de realizar busca local.

Tabela 2. Resultado do ajuste dos parâmetros α e λ (problemas com 100 dimensões).

		Rastringin	Griewank	Sphere	Ackley
α	λ	Média melhor ± desvio padrão			
0,900	0,100	0,045 ± 0,01	0,000778 ± 0,0001	0,091 ± 0,03	0,041 ± 0,00
0,900	0,010	0,044 ± 0,01	0,000779 ± 0,0002	0,079 ± 0,01	0,040 ± 0,01
0,900	0,001	0,050 ± 0,01	0,000874 ± 0,0002	0,090 ± 0,02	0,045 ± 0,01
0,990	0,100	0,290 ± 0,04	0,000822 ± 0,0002	0,122 ± 0,03	0,057 ± 0,01
0,990	0,010	0,304 ± 0,03	0,000845 ± 0,0001	0,121 ± 0,02	0,056 ± 0,00
0,990	0,001	0,310 ± 0,04	0,000926 ± 0,0002	0,135 ± 0,03	0,059 ± 0,01
0,999	0,100	148,886 ± 11,19	0,001465 ± 0,0002	1,038 ± 0,15	1,598 ± 4,21
0,999	0,010	154,626 ± 15,34	0,001431 ± 0,0003	1,116 ± 0,13	0,648 ± 0,04
0,999	0,001	149,444 ± 17,68	0,001541 ± 0,0003	1,153 ± 0,13	0,638 ± 0,06
0,100	0,100	0,042 ± 0,01	**0,000704 ± 0,0001**	0,083 ± 0,02	0,041 ± 0,00
0,100	0,500	0,041 ± 0,01	0,000723 ± 0,0001	0,078 ± 0,01	0,041 ± 0,00
0,100	0,900	0,042 ± 0,00	0,000783 ± 0,0002	0,085 ± 0,02	**0,038 ± 0,00**
0,500	0,100	**0,039 ± 0,00**	0,000788 ± 0,0002	0,081 ± 0,02	0,039 ± 0,01
0,500	0,500	0,044 ± 0,00	0,000745 ± 0,0001	0,078 ± 0,01	0,039 ± 0,00
0,500	0,900	0,042 ± 0,01	0,000799 ± 0,0002	0,091 ± 0,02	0,041 ± 0,00
0,900	0,500	0,041 ± 0,00	0,000708 ± 0,0002	**0,077 ± 0,02**	0,041 ± 0,01
0,900	0,900	0,042 ± 0,00	0,000785 ± 0,0002	0,087 ± 0,02	0,039 ± 0,00

A variação da amplitude é semelhante à variação de temperatura encontrada no algoritmo de resfriamento simulado (*simulated annealing*) [Aarts e Korst 1989]. Se A for muito alto ($A \rightarrow 1$), então a probabilidade de aceitar as novas soluções aumenta. Se A for muito baixo ($A \rightarrow 0$), então uma solução ruim raramente será aceita, diminuindo a diversidade de indivíduos na população. Quando $A \rightarrow 0$ apenas soluções boas serão

Tabela 3. Resultado do ajuste dos parâmetros α e λ (problemas com 200 dimensões).

		Rastringin	Griewank	Sphere	Ackley
α	λ	Média melhor ± desvio padrão			
0,900	0,100	0,343 ± 0,04	0,0058 ± 0,001	0,670 ± 0,09	0,088 ± 0,01
0,900	0,010	0,320 ± 0,05	0,0063 ± 0,001	0,685 ± 0,11	0,092 ± 0,01
0,900	0,001	0,414 ± 0,06	0,0069 ± 0,001	0,719 ± 0,13	0,097 ± 0,01
0,990	0,100	1,007 ± 0,11	0,0067 ± 0,001	0,887 ± 0,13	0,115 ± 0,01
0,990	0,010	1,061 ± 0,13	0,0065 ± 0,001	0,901 ± 0,12	0,115 ± 0,01
0,990	0,001	1,143 ± 0,13	0,0080 ± 0,001	0,960 ± 0,12	0,123 ± 0,01
0,999	0,100	537,306 ± 39,52	0,0096 ± 0,002	5,323 ± 0,54	2,486 ± 4,01
0,999	0,010	525,807 ± 34,09	0,0099 ± 0,001	5,552 ± 0,42	1,620 ± 0,22
0,999	0,001	457,086 ± 87,32	0,0107 ± 0,001	5,626 ± 0,45	1,665 ± 0,25
0,10	0,10	0,334 ± 0,04	0,0059 ± 0,001	0,651 ± 0,10	0,089 ± 0,01
0,10	0,50	0,344 ± 0,05	0,0063 ± 0,001	0,639 ± 0,09	0,087 ± 0,00
0,10	0,90	0,325 ± 0,05	0,0061 ± 0,001	0,647 ± 0,14	0,089 ± 0,00
0,50	0,10	0,348 ± 0,06	**0,0057 ± 0,000**	**0,632 ± 0,08**	**0,086 ± 0,01**
0,50	0,50	0,325 ± 0,04	0,0061 ± 0,001	0,661 ± 0,09	0,089 ± 0,00
0,50	0,90	0,330 ± 0,06	0,0060 ± 0,000	0,651 ± 0,10	0,088 ± 0,01
0,90	0,50	0,340 ± 0,04	0,0058 ± 0,000	0,638 ± 0,11	0,088 ± 0,00
0,90	0,90	**0,319 ± 0,05**	0,0060 ± 0,000	0,641 ± 0,11	0,091 ± 0,00

aceitas, e nessa etapa do BA se transforma em um algoritmo de subida de montanha, podendo ficar preso em um mínimo local.

Se r for muito alto ($r \to 1$), então a probabilidade de realizar busca local aumenta, diminuindo a busca exploratória. Se r for muito baixo ($r \to 0$), então a busca local raramente será realizada.

[Yang 2010b] propôs os seguintes valores padrão: $\alpha = 0,9$ e $\lambda = 0,9$. No entanto, com base nos resultados dos experimentos aqui realizados, a melhor configuração encontrada foi $\alpha = 0,5$ e $\lambda = 0,1$. Com estes valores, o algoritmo BA obteve as melhores médias para as funções Griewank, Sphere e Ackley, todas com 200 dimensões. Por serem problemas mais difíceis do que os semelhantes com 100 dimensões, estes valores se tornam os novos valores padrão para os parâmetros e serão utilizados nos próximos experimentos.

5.2. Comparação do BA com Outros Algoritmos

As Tabelas 4 e 5 mostram a comparação do desempenho do BA (utilizando $\alpha = 0,5$ e $\lambda = 0,1$) com outros algoritmos. Tendo em conta que este trabalho utiliza as versões canônicas (*standard*) dos algoritmos, é provável que melhores resultados possam ser obtidos por ajuste dos seus parâmetros de controle, porém isto está fora do escopo deste trabalho.

A comparação foi realizada através da média da melhor solução sobre todas as execuções independentes. Para as funções Rastrigin e Ackley (100 e 200 dimensões), o BA teve um desempenho muito superior ao PSO, DE e BAS. Para a função Griewank (200 dimensões) o desempenho foi superior ao PSO e BAS e para a função Sphere (100 e 200 dimensões) foi superior apenas ao BAS.

Foi realizada a análise estatística das médias utilizando o teste t de Student [Spiegel 1993]. Foi possível concluir com 97,5% de confiança (ou uma chance de erro de 2,5%) que a diferença das médias entre os algoritmos é significativa.

Tabela 4. Comparação do BA com outros algoritmos – 100 dimensões

	PSO	DE	ABC	BAS	BA
Rastringin	188,10 ± 30,62	57,16 ± 8,94	4,21E-09 ± 1,27E-08	1.249,47 ± 93,00	0,039 ± 0,00
Griewank	0 ± 0	0 ± 0	0 ± 0	1.160,06 ± 195,40	7,88E-04 ± 2,00E-04
Sphere	0 ± 0	0 ± 0	0 ± 0	1,48E+05 ± 5,68E+04	0,081 ± 0,02
Ackley	2,81 ± 0,64	19,95 ± 0,05	0 ± 0	19,93 ± 0,27	0,039 ± 0,01

Tabela 5. Comparação do BA com outros algoritmos – 200 dimensões

	PSO	DE	ABC	BAS	BA
Rastringin	421,71 ± 46,27	211,08 ± 17,68	0,27 ± 0,45	2573,98 ± 149,32	0,348 ± 0,06
Griewank	0,07 ± 0,10	0 ± 0	0 ± 0	2900,69 ± 1305,67	0,0057 ± 0,00
Sphere	0 ± 0	0 ± 0	0 ± 0	3,13E+05 ± 5,03E+04	0,632 ± 0,08
Ackley	6,55 ± 0,98	19,97 ± 0,00	2,30E-11 ± 1,00E-11	19,75 ± 0,67	0,086 ± 0,01

6. Conclusões e Trabalhos Futuros

Uma descrição formal de um algoritmo recente de otimização inspirado na natureza, o BA, foi apresentada neste artigo. A partir desta formalização é possível reproduzir facilmente o algoritmo, o que até então não era possível pois as poucas publicações sobre o BA não continham detalhes suficientes.

Experimentos foram realizados utilizando quatro funções matemáticas. Eles foram realizados com um número propositalmente elevado de dimensões, de modo a transformar o problema em um desafio para todos os algoritmos comparados nos experimentos.

Foram encontrados valores padrão para os principais parâmetros do BA ($\alpha = 0,5$ e $\lambda = 0,1$), valores estes que diferem daqueles propostos pelo criador do algoritmo.

Posteriormente, o BA foi comparado com outros algoritmos populacionais de computação evolucionária. A análise estatística dos resultados experimentais indicaram que a qualidade das soluções obtidas pelo BA supera outros algoritmos existentes como o PSO e, de maneira geral, é competitivo com os demais. O principal motivo do desempenho do BA vem do bom equilíbrio entre a exploração e a intensificação, controlada pelos parâmetros α e λ. Tal equilíbrio é fundamental para qualquer algoritmo de computação evolucionária realizar uma varredura eficiente do espaço de busca e encontrar soluções de boa qualidade. No caso do BA, além dos dois parâmetros citados, também tem o tamanho da população. O fato de ter poucos parâmetros torna um algoritmo menos complexo e potencialmente mais genérico, requisitando menos ajustes para aplicações do mundo real.

O BA também foi comparado com outra versão mais simples, disponibilizada pelo criador [Yang 2010b]. De acordo os experimentos realizados, o BA implementado neste trabalho, que possui o controle contínuo de exploração e intensificação, supera a versão do BA simples.

Futuramente serão estudadas novas técnicas de controle automático de exploração e intensificação do BA e também será aplicado o algoritmo para outros problemas interessantes do mundo real.

Referências

Aarts, E. e Korst, J. (1989). *Simulated Annealing an Boltzmann Machines*. Wiley-Interscience, Chichester, UK.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI.

Karaboga, D. e Akay, B. (2009). A comparative study of artificial bee colony algorithm. *Applied Mathematics and Computation*, 214:108–132.

Kennedy, J. e Eberhart, R. (1995). Particle swarm optimization. páginas 1942–1948, Piscataway, NJ. IEEE Press.

Parpinelli, R. S. e Lopes, H. S. (2011). New inspirations in swarm intelligence: a survey. *International Journal of Bio-Inspired Computation*, 3(1):1–16.

Pham, D. T. e Karaboga, D. (1991). Optimum design of fuzzy logic controllers using genetic algorithms. *Journal of Systems Engineering*, 1:114–118.

Spiegel, M. R. (1993). *Estatística*. Makron Books, São Paulo, SP, 3th edition.

Storn, R. e Price, K. (1997). Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11:341–359.

Vesterstrom, J. e Thomsen, R. (2004). A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In *Proceedings of IEEE Congress on Evolutionary Computation*, volume 3, páginas 1980–1987, Piscataway, NJ. IEEE Press.

Yang, X. S. (2010a). *Nature-Inspired Metaheuristic Algorithms*. Luniver Press, Frome, UK, 2th edition.

Yang, X. S. (2010b). A new metaheuristic bat-inspired algorithm. In Gonzalez, J. R., editor, *Nature Inspired Cooperative Strategies for Optimization*, Studies in Computational Intelligence, páginas 65–74. Springer-Verlag, Berlin, Germany.