OTIMIZAÇÃO DE FUNÇÕES MULTIMODAIS VIA TÉCNICA DE INTELIGÊNCIA COMPUTACIONAL BASEADA EM COLÔNIA DE VAGA-LUMES

Deylon C. F. Couto

Carlos A. Silva

deyloncarlo@gmail.com

carlos.silva@ifmg.edu.br

Instituto Federal de Minas Gerais

Av. Serra da Piedade nº 299 - Morada da Serra, 34515-640, MG, Sabará, Brasil

Lillia S. Barsante

lilliabarsante@gmail.com

Centro Federal de Educação Tecnológica de Minas Gerais

Av. Amazonas nº 7675 - Nova Gameleira, 30510-000, MG, Belo Horizonte, Brasil

Resumo. *Problemas de otimização são comumente encontrados em aplicações práticas de grande relevância econômica e/ou social, como, quando se deseja determinar o maior nível de produção de uma indústria, a quantidade mínima de leitos de um hospital, entre outros problemas na área de administração, economia e engenharias. Muitos desses problemas apresentam um grande número de variáveis e/ou restrições, tornando inviável a solução por meio de métodos exatos. Desta forma, heurísticas computacionais vêm ganhando espaço no tratamento destes problemas. Neste artigo aplicamos o algoritmo de colônia de vaga-lume, proposto na literatura, para otimizar clássicas funções N-dimensionais multimodais. Apresentamos um benchmark entre as funções teste, incluindo novas funções que não foram encontradas em trabalhos da literatura, de acordo com pesquisa bibliográfica realizada, a fim de analisar o desempenho do algoritmo na otimização destas classes de funções, possibilitando concluir a respeito da influência dos principais parâmetros do método computacional nos resultados obtidos. Os resultados mostram que o algoritmo consegue encontrar os ótimos locais em tempo computacional razoável, além de superar o resultado da literatura para algumas funções.*

Palavras-chave: *Inteligência computacional, Otimização, Colônia de vaga-lumes*

1 INTRODUÇÃO

A inteligência computacional (IC) tem sido bastante utilizada para resolver diversos problemas práticos reais. Muitos algoritmos de IC tem surgido nos últimos anos, especialmente os algoritmos baseados em inteligência por enxames. Dentre os algoritmos mais recentes podemos citar: colônia de vaga-lumes (Yang, 2008), colônia de bactérias (Niu & Wang, 2013), otimizador da formiga-leão (Mirjalili, 2015), entre outros. Desde sua criação em 2008, o algoritmo de colônia de vaga-lumes (ACV) vem sendo explorado na otimização de problemas reais como na reconfiguração de antenas para telecomunicação (Chatterjee et al., 2012), no sequenciamento de tarefas (Marichelvam et al., 2014) e outras aplicações.

Neste artigo implementamos e analisamos o desempenho do ACV, proposto por (Yang, 2008), em um *benchmark* de funções multimodais e multidimensionais, visto que estas características são frequentes em funções objetivo de problemas de otimização, especialmente os que retratam aplicações reais. O ACV é baseado no comportamento dos vaga-lumes, sendo estes insetos conhecidos pela emissão de luz (bioluminescência) produzindo pequenos flashes rítmicos luminosos. Estas "luzes" influenciam na atração entre as espécies para fins de reprodução e na atração de presas ou prevenção de predadores. Em relação a um ponto fixo, a intensidade da luz emitida por um vaga-lume diminui à medida que este se afasta do ponto fixo, ou seja, a intensidade da luz é inversamente proporcional à distância. Devido a absorção da luz pelo ar, costuma-se considerar que $I \propto 1/r^2$, onde I é a intensidade luminosa de um vaga-lume e r é a distância entre dois vaga-lumes. No ACV, a intensidade luminosidade é associada à função objetivo a ser otimizada.

Este trabalho está dividido da seguinte forma. Na seção 2 apresentamos a classe de funções que será a base da simulação para o algoritmo ACV. Estas funções serão apresentadas de acordo com sua dimensionalidade. Na seção 3 apresentamos as principais características do algoritmo implementado, bem como o seu pseudocódigo. O *benchmark* proposto neste artigo é descrito na seção 4, onde também serão apresentados os resultados computacionais das simulações do algoritmo, além de uma análise e discussão destes resultados. Por fim, concluímos o trabalho na seção 5, indicando os próximos passos nesta proposta de otimização de funções via métodos de inteligência computacional.

2 FUNÇÕES TESTE MULTIMODAIS

Em problemas de otimização, um interesse comum é encontrar o mínimo (ou máximo) global de funções objetivo. No contexto algorítmico, estas funções representam indicadores de desempenho para avaliar a qualidade das soluções encontradas pelo método computacional, sempre visando a busca da melhor solução. Muitos destes problemas são classificados como NP-Difícil, apresentando uma grande dificuldade em encontrar suas soluções ótimas. Nos últimos anos, muitos algoritmos baseados em inteligência computacional têm sido desenvolvidos para resolver essa classe de problemas. Apesar destes métodos, em geral, não garantirem a otimalidade global da solução, as soluções geralmente são de boa qualidade, representando ótimos locais.

A quantidade de ótimos locais das funções objetivo pode contribuir com o incremento do grau de dificuldade para a resolução do problema e também na escolha do método computacional a ser utilizado. Uma função é considerada multimodal se possui dois ou mais ótimos locais.

Quando se deseja realizar um comparativo de performance entre métodos computacionais aplicados a um certo número de funções multimodais é costume considerar outras características destas funções como a separabilidade e a regularidade. Uma função de p variáveis é dita separável se ela pode ser escrita como a soma de p funções de uma variável. A separabilidade tem a ver com a relação entre as variáveis do problema, sendo esta característica frequentemente utilizada quando se usa métodos de computação evolucionária. Uma função é dita regular se é diferenciável em cada ponto do seu domínio. Funções não-separáveis são mais difíceis para otimizar, aumentando o grau de dificuldade se a função for multimodal. Neste trabalho, todas as funções são não-separáveis, com exceção de *Alpine* e *Alpine02*.

Na literatura encontramos diversos trabalhos abordando problemas de otimização que apresentam funções objetivo multimodais, sobretudo trabalhos cujo tema principal são estudos e *benchmarks* de funções multimodais, como em (Im et al., 2004), (Li et al., 2013) e (Cuevas et al., 2013). Nas próximas seções são apresentadas classes de funções multimodais de acordo com sua dimensionalidade.

2.1 Funções 1D e 2D

Funções unidimensional (1D) e bidimensional (2D) são frequentemente utilizados na modelagem de problemas de otimização. Mesmo que a dimensionalidade da função seja pequena, a complexidade do problema a qual a função está associada, pode ser elevada. Para a classe de problemas de otimização com função objetivo 1D, podemos citar a função *Equal Maxima* utilizada em (Fieldsend, 2013).

Existem clássicos problemas de otimização que utilizam funções 2D, como é o caso de determinar as dimensões de um prisma retangular para que este possua volume máximo ou mínimo. Estes problemas, em geral, envolvem a determinação de derivadas de ordem um e dois, além de que, muitas vezes não apresentam a característica de multimodalidade. Em (Kimura et al., 2013) é feita a inferência de modelos de Vohradsky de redes genéticas otimizando funções multimodais 2D e em (Cuevas & Orta, 2014) é apresentado o algoritmo de busca do Cuco na otimização multimodal, onde é ilustrado um problema de otimização que envolve uma função 2D.

Apesar do objetivo deste trabalho ser analisar o comportamento do ACV na otimização de funções multidimensionais e multimodais, utilizamos duas funções 2D, *Alpine02* e *Bird*, para ilustrar o comportamento do algoritmo em funções visualizáveis. As configurações destas funções são apresentadas na Fig. 1, respectivamente por 1 e 2, sendo descritas sua formulação matemática, a região de factibilidade, o ponto de otimalidade e o valor ótimo da função, sempre considerando a otimalidade como um problema de minimização.

1	$f_1(x) = \prod_{i=1}^{N} \sqrt{x_i} sen(x_i)$	2	$f_2(x) = (x_1 - x_2)^2 + e^{(1-sen(x_1))^2} cos(x_2) + e^{(1-cos(x))^2} sen(x_1)$
	$0 \leq x_1, x_2 \leq 10$		$-2\pi \leq x_1, x_2 \leq 2\pi$
	$x^* = (7.917, 7.917)$		$x^* = (4.701055751981055, 3.15294601960139)$
	$f(x^*) = -6.1295$		$f(x^*) = -106.7645367198034$

Figura 1: Configuração das funções 2D

A visualização das funções 2D pode ser vista na Fig. 2.

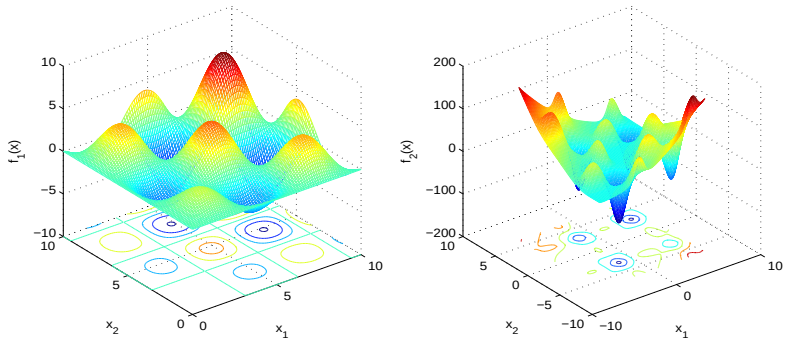


Figura 2: Representação das funções *Alpine02* e *Bird*

2.2 Funções multidimensionais

Em problemas de otimização com funções multidimensionais e multimodais, é costume a utilização de algoritmos baseados em inteligência computacional, como algoritmos genéticos, redes neurais e metaheurísticas em geral, devido à dificuldade de resolução dos mesmos. Na literatura encontram-se trabalhos que tratam de problemas de otimização tendo funções monobjetivo ou multiobjetivo com as características de multidimensionalidade e multimodalidade. O problema de controle e estabilidade de sistemas dinâmicos descrito em (Silva et al., 2011) apresenta uma função objetivo multidimensional e com uma complexa estrutura multimodal, não tendo atualmente uma teoria ou algoritmo que garanta a otimalidade do problema.

Para funções tridimensionais (3D) utilizamos as seguintes funções:*BoxBetts, Gulf, Mishra09* e *SchmidtVetters*, para compor o *benchmark* proposto. Estas funções são apresentadas na Fig. 3, respectivamente por 1, 2, 3 e 4, bem como sua formulação matemática, a região de factibilidade, o ponto de otimalidade e o valor ótimo, sempre considerando a otimalidade como um problema de minimização.

1	$f(x) = \sum_{i=1}^{10} g_i(x)^2$ $g_i(x) = e^{-0.1(i+1)x_1} - e^{-0.1(i+1)x_2}$ $- \left[e^{-0.1(i+1)} - e^{-(i+1)}x_3\right]$ $0.9 \leq x_1, x_3 \leq 1.2, \;\; 9 \leq x_2 \leq 11.2$ $x^* = (1, 10, 1), \;\; f(x^*) = 0$ $x^* = (1, 2, 3), \;\; f(x^*) = 0$	3	$f(x) = \left[ab^2c + abc^2 + b^2 + (x_1 + x_2 - x_3)^2\right]^2$ $a = 2x_1^3 + 5x_1x_2 + 4x_3 - 2x_1^2x_3 - 18$ $b = x_1 + x_2^3 + x_1x_3^2 - 22$ $c = 8x_1^2 + 2x_2x_3 + 2x_2^2 + 3x_2^3 - 52$ $-10 \leq x_1, x_2, x_3 \leq 10$
2	$f(x) = \sum_{i=1}^{N} \left(e^{-\frac{\lvert y_i - x_2\rvert^{x_3}}{x_1}} - t_i\right)$ $t_i = i/100, y_i = 25 + \left[-50log(t_i)\right]^{2/3}$ $0 \leq x_1, x_2, x_3 \leq 60$ $x^* = (50, 25, 1.5), \;\; f(x^*) = 0$	4	$f(x) = \frac{1}{1+(x_1-x_2)^2} + sen\left(\frac{\pi x_2 + x_3}{2}\right) + e^{\left(\frac{x_1+x_2}{x_2} - 2\right)^2}$ $0 \leq x_1, x_2, x_3 \leq 10$ $x_i^* = 0.78547, \;\; f(x^*) = 3$

Figura 3: Configuração das funções 3D

Para se ter uma ideia da complexidade de problemas com um grande número de variáveis, (Molga & Smutnicki, 2005) afirma que a menor instância conhecida do problema de sequencia-

mento de tarefas consiste em 10 tarefas, 10 máquinas e 100 operações, tendo um espaço solução constituído de 10^{48} soluções factíveis, com cada solução possuindo dimensão 90. A resolução do problema percorrendo todas as soluções, gastaria em torno de 25 anos considerando a evolução da computação do ano de 2005. Atualmente é possível encontrarmos funções com mais de 1900 dimensões para esta classe de problemas.

As funções multidimensionais ($N > 3$) utilizadas neste trabalho para a construção do *benchmark* são as seguintes: *Colville*, *Paviani*, *Zahkarov*, *ANNsXOR*, *Salomon*, *Sargan*, *Alpine* e *Cola*. Algumas destas funções foram escolhidas por serem comumente utilizadas em *benchmarks* de funções teste para otimização e outras funções foram escolhidas pela consideração contrária, ou seja, seriam novidades em *benchmarks* desta natureza para o método computacional utilizado. A Figura 4 apresenta a descrição das funções teste N-dimensionais, respectivamente em 1, 2, 3, 4, 5, 6, 7 e 8.

1	$f(x) = 100(x_1 - x_2^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2$ $+(1 - x_3)^2 + 10.1((x_2 - 1)^2 + (x_4 - 1)^2)$ $+19.8(x_2 - 1)(x_4 - 1)$ $N = 4$ $-10 \leq x_i \leq 10,\ i = 1, 2, \ldots, N$ $x^* = (1, \ldots, 1),\ \ f(x^*) = 0$	5	$f(x) = 1 - cos\left(2\pi\sqrt{\sum_{i=1}^{N} x_i^2}\right)$ $+0.1\sqrt{\sum_{i=1}^{N} x_i^2}$ $N = 5$ $-100 \leq x_i \leq 100,\ i = 1, 2, \ldots, N$ $x^* = (0, \ldots, 0),\ \ f(x^*) = 0$
2	$f(x) = \sum_{i=1}^{10} (ln(x_i - 2))^2 + (ln(10 - x_i))^2 - \left(\prod_{i=1}^{10} x_i\right)^{0.2}$ $N = 10$ $2.0001 \leq x_i \leq 10,\ i = 1, 2, \ldots, N$ $x^* = (9.351, \ldots, 9.351),\ \ f(x^*) \approx -45.778$	6	$f(x) = \sum_{i}^{N} \left(x_i^2 + 0.4 \sum_{j \neq 1}^{N} x_i x_j\right)$ N=10 $-100 \leq x_i \leq 100,\ i = 1, 2, \ldots, N$ $x^* = (0, \ldots, 0),\ \ f(x^*) = 0$
3	$f(x) = \sum_{i=1}^{n} x_i^2 + \left(\frac{1}{2}\sum_{i=1}^{n} i x_i\right)^2 + \left(\frac{1}{2}\sum_{i=1}^{n} i x_i\right)^4$ $N = 5$ $-5 \leq x_i \leq 10,\ \ i = 1, 2, \ldots, 5$ $x^* = (0, \ldots, 0),\ \ f(x^*) = 0$	7	$f(x) = \sum_{i=1}^{N} \lvert x_i sen(x_i) + 0.1 x_i \rvert$ $N = 10$ $-10 \leq x_i \leq 10,\ i = 1, 2, \ldots, N$ $x^* = (0, \ldots, 0),\ \ f(x^*) = 0$
4	$f(x) = \sum_{i=1}^{4} f_i(x)$ $f_1(x) = \left(1 + e^{-\frac{x_7}{1+e^{-(x_1+x_2+x_5)}} - \frac{x_8}{1+e^{-(x_3+x_4+x_6)}} - x_9}\right)^{-2}$ $f_2(x) = \left(1 + e^{-\frac{x_7}{1+e^{-x_5}} - \frac{x_8}{1+e^{-x_6}} - x_9}\right)^{-2}$ $f_3(x) = \left(1 - \frac{1}{1+e^{-\frac{x_7}{1+e^{-(x_1+x+5)}} - \frac{x_8}{1+e^{-(x_3+x_6)}} - x_9}}\right)^2$ $f_4(x) = \left(1 - \frac{1}{1+e^{-\frac{x_7}{1+e^{-(x_2+x+5)}} - \frac{x_8}{1+e^{-(x_4+x_6)}} - x_9}}\right)^2$ $N = 9$ $-1 \leq x_i \leq 1,\ i = 1, 2, \ldots, N$ $x^* \approx (0.99999, 0.99993, -0.89414, 0.99994, 0.55932,$ $0.99994, 0.99994, -0.99963, -0.08272),\ \ f(x^*) = 0.959759$	8	$f(N, u) = h(x, y) = \sum_{j<i} (r_{i,j} - d_{i,j})^2$ $x_0 = y_0, x_1 = u_0, x_i = u_{2(i-2)}, y_i = u_{2(i-2)+1}$ $r_{i,j} = \left[(x_i - x_j)^2 + (y_- y_j)^2\right]^{1/2}$ $N = 17$ $0 \leq u_1 \leq 4,\ -4 \leq u_i \leq 4,\ i = 2, \ldots, N$ $u^* = (0.651906, 1.30194, 0.099242, -0.883791,$ $-0.8796, 0.204651, -3.28414, 0.851188, -3.46245,$ $2.53245, -0.895246, 1.40992, -3.07367, 1.96257,$ $-2.97872, -0.807849, -1.68978),\ \ f(u^*) = 11.7464$

Figura 4: Configurações das funções ND

Na função *Cola*, consideramos a matriz simétrica d como:

$$d = \begin{pmatrix} 1 & & & & & & & & & \\ 1.27 & 1 & & & & & & & & \\ 1.69 & 1.43 & 1 & & & & & & & \\ 2.04 & 2.35 & 2.43 & 1 & & & & & & \\ 3.09 & 3.18 & 3.26 & 2.85 & 1 & & & & & \\ 3.20 & 3.22 & 3.27 & 2.88 & 1.55 & 1 & & & & \\ 2.86 & 2.56 & 2.58 & 2.59 & 3.12 & 3.06 & 1 & & & \\ 3.17 & 3.18 & 3.18 & 3.12 & 1.31 & 1.64 & 3.00 & 1 & & \\ 3.21 & 3.18 & 3.18 & 3.17 & 1.70 & 1.36 & 2.95 & 1.32 & 1 & \\ 2.38 & 2.31 & 2.42 & 1.94 & 2.85 & 2.81 & 2.56 & 2.91 & 2.97 & 1 \end{pmatrix}$$

3 ALGORITMO COLÔNIA DE VAGA-LUMES

O algoritmo proposto por (Yang, 2008) é baseado no comportamento social dos vaga-lumes e considera três regras:

- Todos os vaga-lumes são unissex, ou seja, qualquer vaga-lume pode ser atraído por outro. No contexto computacional, qualquer agente (vaga-lume) é influenciado pelo parâmetro de atratividade, não havendo diferença entre os agentes do algoritmo.
- A atratividade é proporcional à luminosidade, ou seja, o vaga-lume com menor "brilho" será atraído pelo vaga-lume de maior "brilho", sendo que esta luminosidade decai com o aumento da distância entre os vaga-lumes. Se não houver brilho, então o vaga-lume move-se aleatoriamente.
- A luminosidade de um vaga-lume é determinada pela função objetivo.

O movimento dos vaga-lumes corresponde ao movimento das soluções no espaço de busca do problema, o qual é guiado por uma função teste dentro de seu domínio. Três parâmetros são essenciais ao funcionamento do método: a atratividade (β), a absorção da luz pelo meio (γ) e a aleatoriedade (α) inerente à movimentação dos vaga-lumes. A variação da atratividade β entre os vaga-lumes a uma distância r é definida como:

$$\beta = \beta_0 e^{-\gamma r^2}, \tag{1}$$

sendo β_0 a atratividade em $r = 0$. A Equação (1) mostra a relação de proporcionalidade entre a atração entre os vaga-lumes e a intensidade luminosa.

O sistema dinâmico que representa a movimentação dos vaga-lumes variando no tempo discreto k é dado por:

$$x_i^{k+1} = x_i^k + \beta_0 e^{-\gamma r_{ij}^2}(x_j^k - x_i^k) + \alpha_k \epsilon_i^k, \tag{2}$$

onde $\alpha \in (0, 1]$, r_{ij} é a distância entre os vaga-lumes x_i e x_j e ϵ_i é um vetor de números aleatórios com distribuição Gaussiana. É fácil perceber duas particularidades em relação aos

parâmetros β e γ. Quando $\beta_0 = 0$ a movimentação dos vaga-lumes segue um passeio aleatório e se $\gamma = 0$ o algoritmo reduz-se ao método de PSO (*Particle Swarm Optimization*). O trabalho de (Yang & He, 2013) aborda sucintamente a configuração de parâmetros do ACV e também apresenta a ordem de complexidade do método.

A complexidade do algoritmo segue a análise apresentada em (Yang & He, 2013). No Algoritmo 1 existem dois *loops* internos, um variando todos os vaga-lumes e fazendo a comparação entre a intensidade luminosa dos pares e o outro *loop* movendo os vaga-lumes de acordo com a Eq. (2). Com uma população de n vaga-lumes, teríamos uma complexidade $O(n^2)$. É possível alterar a implementação do algoritmo para obter uma complexidade $O(nlog(n))$, mas para isso precisaríamos usar um único *loop* para enumerar a atratividade e a luminosidade dos vaga-lumes. Segundo (Yang & He, 2013) a eficiência do ACV se dá pela subdivisão automática do método e a capacidade de tratar multimodalidade, permitindo que os vaga-lumes possam encontrar todos os "ótimos" simultaneamente. Aliado a estas vantagens, os parâmetros do ACV podem ser ajustados para controlar a aleatoriedade de modo a acelerar a convergência do método.

Algoritmo 1: ALGORITMO COLÔNIA DE VAGA-LUMES

Entrada: $n, MaxGen, \beta, \alpha, \gamma$ { número de vaga-lumes, número máximo de gerações, parâmetros de atratividade, aleatoriedade e absorção de luz }
Saída: g^* { melhor solução obtida pelos vaga-lumes}
início
  Determine função objetivo $f(x)$ para $x = (x_1, \ldots, x_d)^T$
  Gere população inicial de vaga-lumes $x^i,\ i = 1, 2, \ldots, n$
  Defina a intensidade de luz como: $\mathrm{I}_i = f(x^i)$
  enquanto *gerações* $\leq MaxGen$ **faça**
    para *todos os vaga-lumes* x_i *e* x_j **faça**
      se $\mathrm{I}_i < \mathrm{I}_j$ **então**
        Mova o vaga-lume x_i para o x_j de acordo com Eq. (2)
      fim
      Varie a atratividade com a distância r via $e^{-\gamma r^2}$
      Calcule novas soluções e atualize a intensidade luminosa
    fim
    Ordene os vaga-lumes e encontre a melhor solução g^*
  fim
fim
retorna g^*

4 *BENCHMARK*

No contexto da otimização existe uma grande variedade de técnicas computacionais, como metaheurísticas, para solucionar problemas de interesse prático. Em virtude desta variedade, é interessante termos informações a respeito da eficiência destas técnicas para aplicá-la a um determinado problema. Neste trabalho buscamos avaliar o desempenho do ACV através do cálculo de mínimos locais (possivelmente globais) de funções multimodais e multidimensionais. Apresentamos o esforço computacional do método através do tempo despendido, a proximidade com a solução ótima, e a variação dos parâmetros do algoritmo visando concluir suas

influências sobre a solução obtida.

Na literatura existem muitos trabalhos que exibem *benchmarks* de métodos computacionais aplicados a funções multimodais e multidimensionais, como em (Cuevas at al., 2013), (Li et al., 2013), (Jamil & Yang, 2013), (Im et al., 2004) e (Dieterich & Hartke ,2012). O trabalho de (Krishnanand e Ghose, 2009) apresenta um *benchmark* de algoritmos baseados em inteligência computacional aplicados a funções multimodais e multidimensionais, incluindo uma versão de algoritmo também baseado no comportamento de vaga-lumes, denominado de *Glow-Worm*, porém apresentando significativas diferenças com o ACV, especialmente no modo da "movimentação" das soluções nas vizinhanças.

(Wolpert & Macready, 1995) mostra que se compararmos dois algoritmos utilizando todas as funções possíveis, o desempenho de quaisquer dos dois algoritmos será em média o mesmo. O importante é selecionar um conjunto de funções com características que permitam concluir sobre a performance do algoritmo. (Eiben & Back, 1997) elabora um conjunto de funções teste, incluindo a função de *Rosenbrock* estendida a p dimensões e a função de *Schwefel* com as características de separabilidade, multimodalidade e regularidade. A multidimensionalidade é uma característica que intensifica a complexidade do problema (Friedman, 1994).

Para as simulações computacionais foi utilizado o software MatLab®R2009a, 64-bit, versão 7.8.0.347, executado em sistema operacional Windows 8.1 de 64 bit, na arquitetura de processador Intel®Core™ i5-4200M CPU 2.50GHz com memória de 4 GB. O *benchmark* é constituído de funções multimodais sendo duas funções de duas dimensões (2D), quatro funções de três dimensões (3D) e oito funções de dimensões variadas (ND), com dimensão maior ou igual a quatro, como apresentadas na seção 2. Para cada função foram realizadas 30 simulações com a configuração de 500 gerações e 50 vaga-lumes. O número de vaga-lumes será justificado na seção 4.1 e o número de gerações consideramos como sendo 10 vezes o valor do número de vaga-lumes. Para a construção do *benchmark* foram considerados 125 variações dos parâmetros α, β e γ conforme apresentado na Tabela 1. Note que apenas o valor inicial de γ

Tabela 1: Configurações dos parâmetros α, β e γ

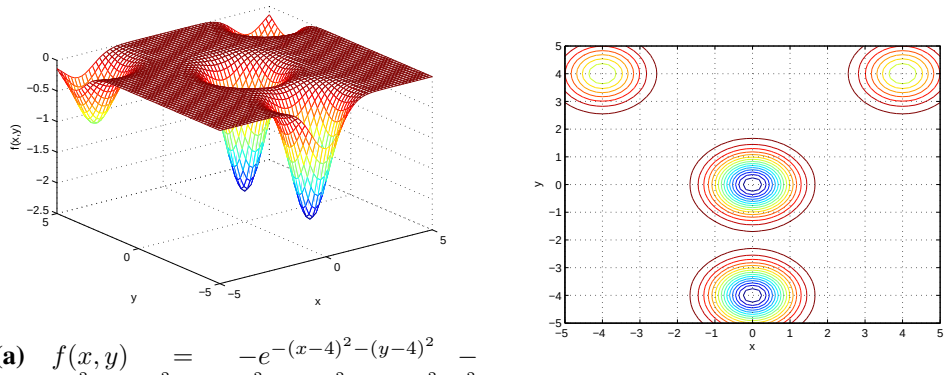
Parâmetros	Configurações				
α	0.1	0.25	0.50	0.75	1.00
β	0.1	0.25	0.50	0.75	1.00
γ	0.01	0.25	0.50	0.75	1.00

difere do padrão dos demais parâmetros, pois queríamos representar uma configuração de uma "quase nula" absorção da luz pelo meio. Esta divisão das configurações representaria casos extremos, bem como medianos e intermediários.

4.1 Análise e discussão dos resultados

Uma das primeiras análises feita neste trabalho foi investigar o impacto da quantidade de vaga-lumes na otimização das funções. Fizemos testes computacionais com as funções 2D, 3D e ND do *benchmark* deste artigo e chegamos ao resultado similar apresentado na Fig. 6. Para clareza de entendimento, utilizaremos uma função 2D multimodal. A função $f(x, y) =$

$-e^{-(x-4)^2-(y-4)^2} - e^{-(x+4)^2-(y-4)^2} - 2e^{-x^2-(y+4)^2} - 2e^{-x^2-y^2}$ possui dois mínimos locais e dois mínimos globais em $(-4,4),(4,4)$ e $(0,-4),(0,0)$, respectivamente, com $f(-4,4) = f(4,4) = -1$ e $f(0,-4) = f(0,0) = -2$. A função e as curvas de nível em torno dos ótimos (locais e globais) são apresentados na Fig. 5.

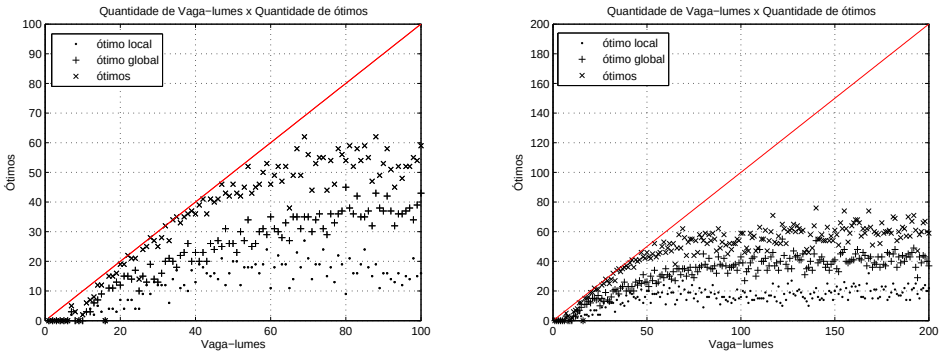


(a) $f(x,y) = -e^{-(x-4)^2-(y-4)^2} - e^{-(x+4)^2-(y-4)^2} - 2e^{-x^2-(y+4)^2} - 2e^{-x^2-y^2}$

(b) Curvas de nível

Figura 5: Ótimos locais e globais

Fixando o número de iterações em $MaxGen = 100$ e variando o número de vaga-lumes, obtemos a relação entre a quantidade de vaga-lumes e as quantidades de ótimos locais, globais e ambos.



(a) Variação de vaga-lumes de 1 a 100 em relação ao número de ótimos (locais+gloais) encontrados

(b) Variação de vaga-lumes de 1 a 200 em relação ao número de ótimos (locais+globais) encontrados

Figura 6: Relação do número de vaga-lumes com o número de ótimos (locais+globais) encontrados, fixando em 100 gerações para cada quantidade de vaga-lumes

Na Fig. 6 é apresentado a eficiência do método ACV aplicado à função da Fig. 5, considerando variações da quantidade de vaga-lumes utilizadas nas simulações e a quantidade destes vaga-lumes que atingiram ótimos locais ou globais. Note que em 20 vaga-lumes, temos cerca de 8 vaga-lumes atingindo ótimos locais, representado por "·", cerca de 11 vaga-lumes atingindo ótimo global, representado por "+", totalizando cerca de 19 vaga-lumes atingindo ótimos (entre locais e globais), representado por "×". Esta análise pode ser estendida à Fig. 5(b), onde poderíamos afirmar que os 8 vaga-lumes atingiram o centro dos círculos superiores da esquerda ou da direita, os 11 vaga-lumes atingiram o centro dos círculos centrais localizados em $(0,-4)$

e $(0, 0)$, totalizando 19 vaga-lumes atingindo algum ótimo, ou seja, um entre os 20 vaga-lumes se desviou da otimalidade. Para esse caso, poderíamos dizer que a eficiência do ACV em função do número de vaga-lumes foi de 95%. Na Fig. 6 traçamos uma linha vermelha (bissetriz) para fins de visualização para identificar a eficiência do algoritmo em função da quantidade de vaga-lumes. No intervalo de 25 a 50 vaga-lumes houve uma melhor aproximação dos pontos do gráfico da Fig. 6 com a bissetriz do primeiro quadrante, ou seja, quase todos os vaga-lumes encontraram o ótimo, seja local ou global. A partir desta estimativa fixamos nossos testes em 50 vaga-lumes.

Nas Tabelas 2, 3 e 4 são apresentadas informações a fim de inferir sobre a eficiência do método computacional utilizado para construir o *benchmark*. Na primeira coluna ("N") é indicado a dimensão da função. Na segunda coluna ("Função") é descrito o nome da função utilizada. Na terceira coluna ("$\overline{f^*} \leq f^*_{\text{LIT}}$ (em %) ") é apresentada a porcentagem das simulações em que o ACV, através do valor ótimo médio obtido pelo algoritmo, atingiu o melhor valor da literatura ou até mesmo foi melhor do que este valor, considerando uma aproximação de 10^{-3}. A quarta coluna ("$f_{\text{MIN}} \leq f^*_{\text{LIT}}$ (em %)") apresenta a porcentagem das simulações em que o ACV, através do valor mínimo obtido pelo algoritmo, atingiu o melhor valor da literatura ou até mesmo foi melhor do que este valor, considerando uma aproximação de 10^{-3}. A informação sobre possíveis resultados que superaram os resultados da literatura está presente na quinta coluna ("$f_{\text{MIN}} < f^*_{\text{LIT}}$"), sendo que "Sim" indica que houve superação de resultado, "Não" não ocorreu superação e "Inconclusivo" indica que não foi possível afirmar com exatidão se houve superação ou não. A maioria das funções tem como fonte o trabalho de (Jamil & Yang, 2013), sendo as exceções comentadas e apresentadas no decorrer da análise dos resultados.

Tabela 2: Eficiência do ACV para funções 2D considerando o valor ótimo médio e o valor mínimo do método $\overline{f^*}$ e f_{MIN}, respectivamente e o melhor valor da literatura f^*_{LIT}

N	Função	$\overline{f^*} \leq f^*_{\text{LIT}}$ (em %)	$f_{\text{MIN}} \leq f^*_{\text{LIT}}$ (em %)	$f_{\text{MIN}} < f^*_{\text{LIT}}$
2	*Alpine02*	96.00	100.00	Inconclusivo
2	*Bird*	61.60	100.00	Sim

Pela Tabela 2 é possível perceber que o ACV conseguiu atingir ou superar o valor ótimo da literatura para todas as duas funções *Alpine02* e *Bird*. O algoritmo teve uma eficiência de 96% e 61.6% para as funções *Alpine02* e *Bird*, respectivamente. Mensuramos esta eficiência como a quantidade, em termos de porcentagem, que o algoritmo, dentre as 30 simulações, atingiu ou superou o melhor valor da literatura. Para a função *Alpine02*, se considerarmos o trabalho de (Jamil & Yang, 2013), o resultado do ACV superou o valor da literatura, porém é fácil perceber que o valor ótimo descrito em (Jamil & Yang, 2013) não está correto para o problema de minimização. Considerando o valor ótimo encontrado em domínio público, o resultado gerado pelo algoritmo é inconclusivo devido a imprecisão das casas decimais do valor disponível na fonte bibliográfica. Para a função *Bird*, nota-se que o ACV teve uma eficiência menor quando comparada à função *Alpine02*, porém, atingiu e superou o resultado da literatura que estava com precisão de treze casas decimais, $f^*_{\text{LIT}} = -106.7645367198034,\ x^* = (4.701055751981055, 3.15294601960139)$, sendo que o algoritmo superou este resultado na oitava casa decimal, $f_{\text{MIN}} = -106.7645367239740,\ x^* = (-1.582130932284172 - 3.130254317326868)$.

Tabela 3: Eficiência do ACV para funções 3D considerando o valor ótimo médio e o valor mínimo do método $\overline{f^*}$ e f_{MIN}, respectivamente e o melhor valor da literatura f^*_{LIT}

N	Função	$\overline{f^*} \leq f^*_{\text{LIT}}$ (em %)	$f_{\text{MIN}} \leq f^*_{\text{LIT}}$ (em %)	$f_{\text{MIN}} < f^*_{\text{LIT}}$
3	*SchmidtVetters*	100.00	100.00	Sim
3	*Mishra09*	94.40	100.00	Não
3	*Gulf*	100.00	100.00	Sim
3	*BoxBetts*	100.00	76.80	Não

Para as funções da Tabela 3, a literatura fornece para a função *SchmidtVetters*, $x^* = (0.78547, 0.78547, 0.78547)$ com $f^*_{\text{LIT}} = 3$, considerando a região $x_i \in [0, 10],\ i = 1, 2, 3$. O ACV retornou $x^* = (7.07081, 10, 3.14\ 1641)$ com $f^* = 0.193972$. Poderíamos supor que o autor da fonte bibliográfica tenha cometido um erro tendo a intenção de maximizar, porém é fácil perceber o impacto de x_2 na otimização da função e sugerir uma configuração que exceda o valor sugerido pelo autor. Encontramos outra formulação desta função em (Gábor et al., 1978) e mesmo para esta outra função, o ACV conseguiu superar o ótimo. Para fins de esclarecimentos e de futuras reproduções de resultados a Eq. (3) descreve a configuração de *SchmidtVetters* de acordo com (Gábor et al., 1978):

$$f(x) = \frac{1}{\left[1-(x_1-x_2)^2\right]^2} + sen(x_2 + x_3 + \pi) + e^{\left(\frac{x_1+x_3}{x_3}-2\right)^2},\ x^* = \left(\tfrac{\pi}{4}, \tfrac{\pi}{4}, \tfrac{\pi}{4}\right),\ \ f^* = 1. \quad (3)$$

O ACV retornou $x^* = (9.99,\ 4.14,\ 9.99)$ com $f^* = 9.31 \times 10^{-4}$, o que é mais de 10^3 vezes melhor. O algoritmo teve um bom desempenho para as funções *Mishra09* e *BoxBetts*, sempre encontrando o valor da literatura da *Mishra09* e na maioria das vezes para a função *BoxBetts*, porém, para nenhuma das duas funções o algoritmo obteve resultado melhor do que da literatura. A função *Gulf* é bastante conhecida em diversas fontes da literatura como *Gulf Research Problem*. Neste trabalho reportamos à função apresentada em (Gavana, 2013) em que utiliza a dimensão do problema como limite do somatório da função. Especificamente para esta configuração, o algoritmo teve um ótimo desempenho superando o valor da literatura, $f_{\text{MIN}} = -0.06$, contra $f^*_{\text{LIT}} = 0$, tendo como argumentos $x^* = (1.0659,\ 13.0799,\ 32.7687)$ e $x^* = (50,\ 25,\ 1.5)$, respectivamente.

Pela Tabela 4 a função *Colville* não apresentou um bom comportamento médio dos valores ótimo obtidos pelo algoritmo, no entanto ele foi capaz de encontrar o valor da literatura no escopo de 30 simulações. Em *Salomon*, há uma indicação que o ACV ficou "preso" ou convergiu a maior parte dos vaga-lumes para um ótimo local, fazendo com que seu desempenho não seja tão bom. Essa indicação é devida a análise do desvio padrão dos valores obtidos pelo ACV em torno de $\tilde{x}$, tal que $f(\tilde{x}) \approx 0.0998$. A função *Zahkarov* tem um desempenho robusto. Para a função *ANNsXOR* o algoritmo atingiu o ótimo, porém, na média não alcançou o valor da literatura. Este alcance dista em média 0.005 do valor da literatura. Em alguns resultados, parece haver uma superação de resultado, mas esta indicação é influenciada por possíveis erros de arredondamento. Similarmente acontece com a função *Paviani* que tem um desempenho razoável e não é possível concluir se há superação de resultado devido a imprecisão das casas decimais fornecidas pela literatura. A função *Alpine* tem um bom desempenho no alcance do ótimo da literatura, mas na média seu desempenho não alcançou 20%. A função *Sargan* teve

Tabela 4: Eficiência do ACV para funções ND considerando o valor ótimo médio e o valor mínimo do método $\overline{f^*}$ e f_{MIN}, respectivamente e o melhor valor da literatura f^*_{LIT}

N	Função	$\overline{f^*} \leq f^*_{\text{LIT}}$ (em %)	$f_{\text{MIN}} \leq f^*_{\text{LIT}}$ (em %)	$f_{\text{MIN}} < f^*_{\text{LIT}}$
4	*Colville*	0.00	100.00	Não
5	*Salomon*	0.00	8.80	Não
5	*Zahkarov*	100.00	100.00	Não
9	*ANNsXOR*	0.00	76.80	Inconclusivo
10	*Alpine*	16.80	76.00	Não
10	*Paviani*	77.60	98.40	Inconclusivo
10	*Sargan*	36.00	57.60	Não
17	*Cola*	0.00	0.80	Não

um desempenho médio para baixo. A função *Cola* foi a função mais complexa, contendo 17 variáveis com vários mínimos. O desempenho não foi bom, porém o algoritmo foi capaz de alcançar o ótimo em pelo menos uma das 30 simulações.

A Tabela 5 quantifica a influência dos parâmetros de aleatoriedade, atratividade e absorção da luz pelo meio na otimização das funções multidimensionais. As colunas "Função" e "N" descrevem a função e sua dimensionalidade, respectivamente. As colunas "α", "β" e "γ" são os parâmetros característicos do algoritmo. As colunas "%" à direita de cada parâmetro indicam a frequência do parâmetro (à esquerda) nas melhores soluções. Na Tabela 5, o símbolo $*$ significa que todas as configurações do parâmetro são válidas, sendo as configurações descritas na Tabela 1. A notação $*/\{c_1, \ldots, c_n\}$ significa que todas as configurações do parâmetro são válidas com exceção das configurações $c_1, \ldots, c_n$.

Tabela 5: Impacto dos parâmetros nas soluções das funções ND

N	Função	α	%	β	%	γ	%
4	Colville	*	20.00	*	20.00	*	20.00
5	Salomon	0.1,0.25	45.45	0.5	54.54	0.75	45.45
5	Zahkarov	*	20.00	*	20.00	*	20.00
9	ANNsXOR	0.75,1	26.04	1	26.04	0.01,0.25	21.87
10	Alpine	1.00	21.05	0.1	26.31	*/{0.01}	21.05
10	Paviani	*/{0.1}	20.32	*/{0.75,1}	20.32	*/{0.25,0.75}	20.32
10	Sargan	0.1,0.25	34.72	*/{0.25,0.5}	20.83	*/{0.01,1}	20.83
17	Cola	-	-	-	-	-	-

Para a função Cola a única configuração válida foi $\alpha = 1,\ \beta = 0.1$ e $\gamma = 0.5$. Usando

como métrica a frequência, é possível concluir pela Tabela 5 que para a otimização das funções N-dimensionais deste artigo, os parâmetros a serem escolhidos seriam: $\alpha = 0.25$, $\beta = 0.1$ e $\gamma = 0.25, 0.5$ ou 0.75.

Foram construídas estruturas computacionais para armazenarem informações essenciais para a análise do ACV mediante às funções do *benchmark*. A Tabela 6 mostra um exemplo destas informações para a configuração de $\alpha = 0.5, \beta = 0.5$ e $\gamma = 0.5$. A coluna N representa a dimensionalidade da função. As variáveis $\overline{f^*}_{\text{ACV}}$ e f_{MIN} representam o valor médio ótimo encontrado pelo ACV usando 30 simulações e o menor valor encontrado pelo método, respectivamente. O desvio padrão dos valores obtidos pelo ACV é representado por $sdf(f)$. A taxa de sucesso (ts) é obtida pela porcentagem de vezes que o método aproxima do valor ótimo da literatura com precisão de 10^{-3}. Nomeamos a variável btf como sendo a porcentagem de vezes que o valor obtido pelo método é menor, ou seja, melhor do que o valor ótimo da literatura. Por fim, o tempo médio despendido pelo algoritmo é representado por $\bar{t}$. Apesar de não constar nesta tabela, foram armazenadas no *benchmark* as informações de $\arg\limits_{x} f^*_{\text{ACV}}(x)$ e $\arg\limits_{x} f_{\text{MIN}}(x)$, a fim de explicitar os valores ótimos encontrados pelo algoritmo na minimização das funções teste.

Tabela 6: *Benchmarck* do ACV para funções multimodais com as configurações

$\alpha = 0.5,\ \beta = 0.5,\ \gamma = 0.5$							
Função	N	$\overline{f^*}_{\text{ACV}}$	f_{MIN}	$std(f)$	ts	btf	$\bar{t}$
Alpine02	2	-6.1295038894	-6.1295038910	0.0000000016	100	100	2.1128
Bird	2	-106.764536605	-106.764536748	1.5334e-007	100	13.33	2.1517
SchmidtVetters	3	0.19397	0.19397	1.5325e-010	0*	100	2.1589
Mishra09	3	1.24405e-004	2.38503e-017	4.7359e-004	93.33	0	2.1800
Gulf	3	-0.0600	-0.0600	0	0*	100	1.3309
BoxBetts	3	3.9028e-010	6.5817e-014	1.6771e-009	100	0	1.5649
Colville	4	0.1435	2.8205e-006	0.6725	56.66	0	2.3075
Salomon	5	0.099873347008446	0.099873345846811	0.000000001	0	0	2.2171
Zahkarov	5	2.2535e-007	5.3216e-008	0.00000013	100	0	2.2626
ANNsXOR	9	0.973490	0.959774	0.0071	6.66	0	2.3302
Alpine	10	0.0014	1.9254e-004	0.00120	50	0	2.2458
Paviani	10	-45.778468865140418	-45.778469394843334	0.0000003	100	100*	2.3027
Sargan	10	0.0019	4.2477e-004	0.00069	6.66	0	2.2829
Cola	17	13.8453	11.9382	1.4412	0	0	2.7759

Em *Bird*, o ACV apresentou 13.33% de vezes em que obteve resultado melhor que o ótimo da literatura, dentre as 30 simulações. Note que na Tabela 2 aparece o valor de 100% para os valores menores ou iguais que o ótimo da literatura, porém, esta análise levou em consideração se em cada um dos "blocos" de simulação o algoritmo atingiu pelo menos uma vez o valor ótimo ou melhor que o ótimo da literatura. A taxa de sucesso das funções *SchmidtVetters* e *Gulf* foi de 0%, pois estava sendo considerado o alcance ao valor ótimo conhecido usando uma distância de 10^{-3}, e para estas funções o valor obtido pelo algoritmo foi sempre melhor que o valor da literatura com uma "grande" diferença entre eles. Para a função *Paviani* foi considerada que em 100% das vezes o algoritmo superou o ótimo devido a precisão das casas decimais do valor ótimo conhecido da literatura.

5 CONCLUSÃO

Neste trabalho foi apresentado um *benchmark* de funções multidimensionais e multimodais. As funções foram divididas de acordo com sua dimensionalidade. Procuramos variar os parâmetros do algoritmo em relação à aleatoriedade do método, o parâmetro de atração entre os vaga-lumes e a absorção da luz pelo meio. O algoritmo ACV encontra a solução para as funções em um tempo médio de 2.16 segundos, o qual é considerado um tempo baixo para uma configuração de 50 vaga-lumes e um número máximo de gerações de 500, além de se tratar de um algoritmo de tempo polinomial. Em uma média global, verificou-se que em todas as simulações, pouco mais de 25%, foram obtidos valores ótimos melhores do que da literatura. Isso representa um bom indicativo de eficiência do ACV, pois não era esperado superar estes resultados, mas sim obter soluções próximas ou exatamente estas soluções. Para o conjunto de funções utilizadas no *benchmark* há um indício de que o algoritmo se comporta melhor com uma aleatoriedade média-baixa, em torno de 25%, uma baixa atratividade entre os vaga-lumes, por volta de 10%, e com a absorção da luz dos vaga-lumes pelo meio com valores diferentes dos extremos, ou seja, não assumindo valores muito próximos da total absorção ou da ausência de absorção. Como trabalhos futuros, pretende-se otimizar a configuração de parâmetros e ampliar o *benchmark* para funções restritivas, possibilitando utilizar o algoritmo com eficiência para problemas gerais de otimização.

AGRADECIMENTOS

Agradecemos à Fapemig pelo apoio financeiro despendido ao desenvolvimento deste trabalho.

REFERÊNCIAS

Chatterjee, A., Mahanti, G. K. & Chatterjee, A., 2012. Design of a fully digital controlled reconfigurable switched beam concentric ring array antenna using firefly and particle swarm optimization algorithm. *Progress in Electromagnetic Research B*, vol. 36, pp. 113-131.

Cuevas, E., Zaldívar, D. & Cisneros, M. P., 2013. A swarm optimization algorithm for multimodal functions and is application in multicircle detection. *Mathematical Problems in Enginnering*, vol. 2013, pp. 1-22.

Cuevas, E. & Orta, A. R., 2014. A cuckoo search algorithm for multimodal optimization, *The Scientific World Journal*, vol. 2014, pp. 1-20.

Dieterich, J. M. & Hartke, B., 2012. Empirical review of standard benchmark functions using evolutionary global optimization. *Applied Mathematics*, vol. 3 n. 10A, pp. 1552-1564.

Eiben, A. E. & Bäck, T., 1997. Empirical investigation of multi-parent recombination operators in evolution strategies, *Evolutionary Computation*, vol. 5, n. 3, pp. 347-365.

Fieldsend, J. E., 2013. Multi-Modal Optimisation using a Localised Surrogates Assisted Evolutionary Algorithm. In *13th UK Workshop on Computational Intelligence - UKCI*, vol. 1, pp. 88-95.

Fister Jr., I., Yang, X. S., Fister, I., Brest, J. & Fister, D., 2013. A brief review of nature-inspired algorithms for optmization. *Elektrotehniski Vestnik*, vol. 80, n. 3, pp. 1-7.

Friedman, J. H., An overview of predictive learning and function approximation. In V. Cherkassky, J. H. Friedman, and H. Wechsler, editors, From Statistics to Neural Networks, Theory and Pattern Recognition Applications, volume 136 of NATO ASI Series F, pages 1-61. Springer-Verlag, 1994.

Gábor, T., László & Endre, C., 1978. On the genetic laws of common isolated congenital malformations, *Alkalmazott Matematikai Lapok*, vol. 4, pp. 1-25 (in Hungarian).

Gavana, A., 2013, Test Functions Index, [Acessado em 28 de julho, 2015]. `http://infinity77.net/global_optimization/test_functions.html`

Im, C. H., Kim, H. K., Jung, H. K. & Choi, K., 2004. A novel algorithm for multimodal function optimization based on evolution strategy. *IEEE Transaction on Magnetics*, vol. 40, n. 2, pp. 1224-1227.

Jamil, M. & Yang, X-S., 2013. A literature survey of benchmark functions for global optimization problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 4, n. 2, pp. 150-194.

Kimura, S., Sato, M. & Hatakeyama, M. O., 2013. Inference of vohradský's models of genetic networks by solving two-dimensional function optimization problems. *PloS One*, vol. 8, n. 12, pp. e83308.

Krishnanand, K. N. & D. G. Ghose, 2009. Swarm optimization for simultaneous capture of multiple local optima of multimodal functions. *Swarm Intelligence*, vol. 3, pp. 87-124.

Li, X., Engelbrecht, A. & Epitropakis, M. G., 2013. Benchmark functions for CEC'2013 special session and competition on niching methods for multimodal function optimization. *Technical report, RMIT University, Evolutionary Computation and Machine Learning Group*, Australia.

Marichelvam, M. K., Prabaharan, T. & Yang, X. S., 2014. A discrete firefly algorithm for the multi-objective hybrid flow shop scheduling problems. *IEEE Transactions on Evolutionary Computation*, vol. 18, n. 2, pp. 301-305.

Mirjalili, S., 2015. The ant lion optimizer. *Advances in Engineering Software*, vol. 83, pp. 80-98.

Molga, M. & Smutnicki, C., 2005. Test functions for optimization needs, `http://www.zsd.ict.pwr.wroc.pl/files/docs/functions.pdf`.

Niu, B. & Wang, H., 2013. Bacterial colony optimization. *Discrete Dynamics in Nature and Society*, vol. 2012, pp. 1-28.

Silva, C. A., Bortolin, D. C. & Costa, E. F., 2011. An algorithm for the long run average cost problem for linear systems with indirect observation of Markov jump parameters. In *18th International Federation of Automatic Control - World Congress - IFAC*, vol. 1, pp. 12668-12673.

Yang ,X. S., 2008. *Nature-Inspired Metaheuristic Algorithms*. Addison-Wesley, Luniver Press, UK.

Yang, X. S., He, X., 2013. Firefly algorithm: recent advances and applications. *International Journal of Swarm Intelligence*, vol. 1, n. 1, pp.36-50.

Wingo, D., 1984. Fitting three parameter lognormal model by numerical global optimization-an

improved algorithm. *Computational Statistics and Data Analysis*, vol. 2, n. 1, pp. 13-25.

Wolpert, D. H. & Macready, W. G., 1995. No free-lunch theorems for search. *Technical Report 95-02-010*, Santa Fe Institute.