Um Estudo Sobre Meta-Heurísticas em Otimização Multimodal

Lúcio Flávio de Jesus Silva
Programa de Pós-Graduação em Engenharia da Computação e Sistemas (PECS)
Universidade Estadual do Maranhão (UEMA)
São Luis, Maranhão
lucioslv@hotmail.com.br

Omar Andres Carmona Cortes
Departamento de Computação (DComp)
Instituto Federal do Maranhão (IFMA)
São Luis, Maranhão
omar@ifma.edu.br

RESUMO

This paper presents an investigation of four metaheuristics for multimodal optimization. The algorithms have been implemented in R and compared against each other using well-known benchmark functions. We implemented the following algorithms: Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Differential Evolution (DE), and Bat Algorithm (BA). For comparisons, we used five multimodal benchmarks: Rosenbrock, Griewank, Ackley, Schwefel, and Alpine. Preliminary results show that Bat Algorithm and Genetic Algorithm tend to discover the best solution considering the parameters that have been set up.

KEYWORDS

Meta-Heurísticas, Algoritmo Bat, Multimodal, Otimização

1 INTRODUÇÃO

Problemas de otimização numérica sem restrições são comuns em aplicações práticas e sendo estes formados normalmente por várias variáveis, limita-se o tipo de técnica ou algoritmo que pode ser utilizado para sua solução [1]. Basicamente, otimizar uma função numérica sem restrições é maximizar ou minimizar a função em questão [2]. Matematicamente, deve-se min ou max$f(x)$, na qual $x \in \mathbf{R}^n$ e $n \geq 1$. Assim, a solução $x*$ é uma solução global de minimização se $f(x*) < f(x)\ \forall\ x$. De forma análoga, no caso de maximização $f(x*) > f(x)\ \forall\ x$, sendo x um escalar quando $n = 1$ ou um vetor quando $n > 1$.

Ao otimizar um função numérica, algoritmos clássicos de otimização podem ser utilizados; porém, em problemas com poucas variáveis, já que as técnicas clássicas normalmente exigem que as funções sejam diferenciáveis. Essa característica leva a outro problema que é a quantidade de variáveis, ou seja, técnicas clássicas lidam com problemas com poucas variáveis. Além disso, restrições são difíceis de ser tratadas por estas técnicas.

Nesse contexto, para solucionar problemas de otimização numérica surgem as meta-heurísticas como uma alternativa viável, pois podem lidar com muitas variáveis e também com as restrições impostas pelo problema. Sendo assim, neste trabalho serão utilizadas as seguintes meta-heurísticas: Algoritmo Genéticos (AG) [3], a Otimização por Nuvem de Partículas (PSO) [4], Evolução Diferencial (ED) [5] e *Bat Algorithm* (BA) [6]. Esses algoritmos serão utilizados na resolução de cinco funções de benchmarks apresentadas na Tabela 1 que são funções multimodais, ou seja, que apresentam muitos mínimos locais, representando um desafio para os algoritmos que irão solucioná-las. Dentre as funções apresentadas, a função Rosenbrock é considerada monomodal apenas em duas dimensões, a função generalizada usada neste trabalho pode ser considerada multimodal em dimensões maiores ou iguais a três como pode ser visto no trabalho de Rifaie & Aber [7].

Outra característica importante que deve ser observada nas funções utilizadas neste trabalho é a separabilidade, ou seja, a possibilidade de separar uma equação em equações menores e independentes. No caso de funções não separáveis, existe uma dependência entre as variáveis que não permite a separação, fazendo com que sua otimização seja ainda mais desafiadora. De qualquer forma essas funções são comuns, sendo encontradas em trabalho, como por exemplo, [8], [9] e [10], dentre outros.

Nesse contexto, este artigo esta dividido da seguinte forma: a Seção 2 descreve com detalhes o algoritmo Bat e fornece referências para as demais meta-heuristicas utilizadas; a Seção 3 mostra um experimento preliminar com todos os algoritmos envolvidos; finalmente, a Seção 4 apresenta as conclusões deste trabalho e direcionamentos futuros.

2 META-HEURÍSTICAS

Como mencionado previamente, as meta-heurísticas utilizadas neste trabalho são os Algoritmo Genéticos (AG), a Otimização por Nuvem de Partículas (PSO), a Evolução Diferencial (ED) e o Algoritmo Bat (BA). Por questões de espaço apenas o algoritmo Bat é detalhado, porém as referências fornecidas na introdução são suficientes para encontrar os trabalhos originais que proporam as referidas meta-heurísticas. O código do AG pode ser encontrado no trabalho de Cortes & Silva [1].

O Bat Algorithm é uma meta-heurística inspirada no comportamento dos morcegos, mais precisamente em seu sistema de ecolocalização utilizado para encontrar alimentos. Para localizar o alimento os morcegos emitem um som bastante alto, mas inaudível para os seres humanos, e esperam receber o eco desse pulso de som dos objetos à sua volta. Embora o sistema de ecolocalização seja bastante complexo, algumas simplificações são necessárias para simular seu comportamento através de um algoritmo:

i. morcegos usam a ecolocalização para medir distâncias e eles sabem a diferença entre comida e obstáculos;
ii. morcegos voam aleatoriamente com velocidade v_i, numa posição x_i, com uma frequência fixa f_{min}, variando o comprimento de onda λ e amplitude A_0. Eles podem automaticamente ajustar tanto a frequência emitida quanto ajustar a taxa de emissão $r \in [0, 1]$ dependendo da distância de sua presa;
iii. embora a amplitude possa variar de muitas formas, assume-se que ela pode variar de A_0 (um número grande positivo) até uma constante mínima A_{min}.

Tabela 1: Funções de Benchmark sem Restrições

Nome	Função	Domínio	Mínimo	Separável
Rosenbrock	$f_1(x) = \sum_{i=1}^{n} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	[-5,10]	0	Não
Griewank	$f_2(x) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}})$	[-600,600]	0	Não
Ackley	$f_3(x) = -20\exp(-0.2\sqrt{\frac{1}{n} \sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n} \sum_{i=1}^{n} \cos(2\pi x_i)) + 20 + exp(1)$	[-32,32]	0	Não
Schwefel	$f_4(x) = \sum_{i=1}^{n} -x_i \sin \sqrt{\lvert x_i \rvert}$	[-500,500]	0	Sim
Alpine	$f_5(x) = \sum_{i=1}^{n} \lvert x_i \sin(x_i) + 0.1x_i \rvert$	[0,10]	0	Sim

O Algoritmo 1 apresenta o pseudocódigo do *Bat Algorithm*, sendo que o movimento dos morcegos é dado pela atualização da frequência $f_i = f_{min} + (f_{max} - f_{min})\beta$, na qual $\beta \in [0, 1]$, na atualização da velocidade $v_i^{t+1} = v_i^t + (x_i^t - x*)f_i$, e na atualização de posição através de $x_i^{t+1} = x_i^t + v_i^{t+1}$, sendo $x*$ a melhor posição até o momento. A solução local é gerada por $x_{novo} = x_{atual} + \epsilon A^t$, na qual $\epsilon \; in \; [-1, 1]$ aleatoriamente gerado e A^t é a média da amplitude de todos os morcegos na atual iteração. De maneira geral, o algoritmo BA pode ser considerado como um PSO com busca local intensa [6]. A amplitude A_i, também chamada de ruído ou sonoridade, juntamente com a taxa de emissão de pulsos r_i devem ser também atualizadas a medida que o algoritmo avança. Nesse contexto, A é atualizado por $A_i^{t+1} = \alpha A_i^t$, e r é modificado por $r_i^{t+1} = r_i^0(1 - \exp -\gamma t)$, nas quais α e γ são constantes. Em caso de simplificação pode se utilizar $\alpha = \gamma$. Lembrando que tanto A quanto r são atualizados somente quando uma nova solução é melhorada, significando que o morcego esta se movendo na direção da solução ótima.

Algoritmo 1 - Pseudocódigo do Algoritmo Bat

```
Inicializa população de morcegos x_i e velocidade v_i
Definir aleatoriamente a frequência f_i em x_i
Inicializar taxa de pulsos r_i e amplitude A_i
enquanto (Critério de Parada = FALSO) faça
    se (rand() > r_i) então
        s ← Seleciona solução dentre as melhores
        l ← Gera uma solução local(s)
    fim se
    Gera uma solução voando aleatoriamente
    se (rand() < A_i E f(x_i) < f(x*)) então
        Aceita a nova solução
        Aumentar r_i e Diminuir A_i
    fim se
    Avaliar morcegos e encontrar o melhor x*
fim enquanto
```

Na próxima seção são apresentados os experimentos preliminares comparando o algoritmo Bat (BA) com os demais algoritmos (AG, PSO e ED), usando as funções numéricas multimodais mostradas previamente.

3 EXPERIMENTO

Os experimentos foram realizados no Windows 10, 64 bits, 8 GB de RAM, 240 GB de SSD, linguagem R versão 3.3.3 e RStudio 1.1.463. Foram usadas 1000 iterações, dimensão = 30, população = 50 e 30 execuções para cada algoritmo. Os parâmetros de cada algoritmo foram os seguintes:

i. **GA** - $p_c = 0.6$, $p_m = 0.01$, seleção por torneio com 4 indivíduos, crossover simples, mutação uniforme e sem elitismo;
ii. **PSO** - $w = 0.7$, $c_1 = c_2 = 2$, $V_{min} = -100$ e $V_{max} = 100$;
iii. **DE** - $p_c = 0.7$, $F = 0.4$ e DE/Best/01;
iv. **BA** - $A = 0.5$, $r = 0.5$, $A_{min} = 0$, $A_{max} = 2$.

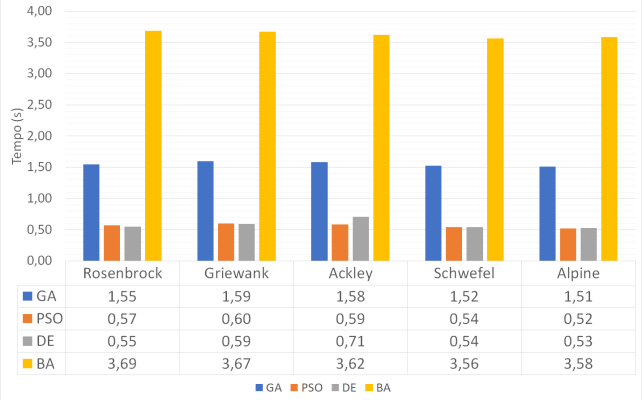
A Tabela 2 mostra os melhores resultados encontrados nas 30 execuções juntamente com os respectivos desvios padrão (D.P), na qual observou-se que nas funções Rosenbrock, Griewank e Alpine, o BA apresentou os melhores resultados. Enquanto que nas funções Ackley e Schwefel o AG apresentou os melhores resultados. Na última linha da tabela apresenta-se o F encontrado em uma Análise de Variância (ANOVA) cujo F_{crtico} é igual a 2,683, indicando que existem diferenças entre os algoritmos nas funções Griewank, Ackley, Schwefel e Alpine. Na função Rosenbrock, a diferença não é estatisticamente significativa, pois o desvio padrão tende a ser alto. No entanto, a melhor solução encontrada na função Rosenbrock aparentemente faz diferença.

Tabela 2: Melhores resultados e desvio padrão

	Rosenbrock	Griewank	Ackley	Schwefel	Alpine
AG	26,8692	2,21e-06	**0,0895**	**1,5083**	0,0142
D.P.	29,8934	2,0567e-06	0,03207	1,4767	0,0076
PSO	131,9989	1,878e-05	0,0974	46,2166	0,0584
D.P.	713,3293	0,00032	0,4878	70,9426	2,2752
ED	612,0827	**0,0005**	2,0810	58,134	0,3571
D.P.	1276,6981	0,0006	0,4172	26,588	1,078
BA	**21,2508**	**2,599e-09**	2,8874	27,8614	**0,0012**
D.P.	36,6211	6,67e-10	0,618	40.698	2,5675
F	1,679	**3,203**	**5,8690**	**2,918**	**4,1666**

Um teste de Tukey indica que não há diferenças entre ED vs BA ($p = 0,99$), AG vs BA ($p = 0,974$) e GA vs DE ($p = 0,98$) em Griewank, o que faz sentido já que as soluções são realmente parecidas. Em Ackley, não há diferenças significativas entre ED vs BA ($p = 0,98$) e PSO vs AG ($p = 0,97$). Em Schwefel, embora a diferença seja bem perceptível, a maior diferença ocorre entre BA vs ED ($p = 0,02$) novamente possivelmente pelo desvio padrão. Finalmente na função Alpine, a maior diferença ocorre entre PSO vs BA ($p = 0.006$), indicando que BA tende a apresentar melhores resultados.

O tempo médio de cada meta-heurística é apresentado na Figura 1, na qual pode-se observar que as meta-heurísticas mais rápidas são



	Rosenbrock	Griewank	Ackley	Schwefel	Alpine
GA	1,55	1,59	1,58	1,52	1,51
PSO	0,57	0,60	0,59	0,54	0,52
DE	0,55	0,59	0,71	0,54	0,53
BA	3,69	3,67	3,62	3,56	3,58

Figura 1: Média dos tempos de execução

a ED e o PSO. Porém, mesmo estando no patamar dos três segundos e um segundo e meio, respectivamente, as soluções alcançadas pelos algoritmos BA e AG tendem a ser bem melhores que os outros algoritmos, especialmente na função Rosenbrock.

4 CONCLUSÕES

Este artigo apresentou um estudo de várias meta-heurísticas utilizadas na otimização numérica de funções de benchmark multimodais bem conhecidas na literatura. De maneira geral o BA e o AG se mostraram superiores aos demais algoritmos apresentados nas 30 execuções, sendo BA melhor nas funções Rosenbrock, Griewank e Alpine, enquanto que o AG foi melhor nas funções Ackley e Schwefel. Em termos de tempo de execução os algoritmos mais lentos são o BA e o AG. Porém, são os algoritmos que tendem a apresentar os melhores resultados.

Trabalhos futuros incluem, mas não estão limitados a: (i) estudar a otimização dos parâmetros utilizados no experimento; (ii) estender os experimentos e análises usando projeto de experimentos e análise fatorial; e (iii) encontrar formas de hibridizar os algoritmos de modo a produzir soluções de maior qualidade; e, (iv) hibridizar os algoritmos de forma adaptativa, ou seja, determinar quais operações de quais algoritmos serão realizadas em tempo de execução.

REFERÊNCIAS

[1] O. A. C. Cortes and J. C. da Silva. Unconstrained numerical optimization using real-coded genetic algorithms: A study case using benchmark functions in R from scratch. 11(3):1–11, Novembro .

[2] J. Nocedal and S Wright, editors. *Numerical Optimization*. Springer, New York - USA, 2006.

[3] Z. Michalewicz, editor. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, New York, 3 edition, 1999.

[4] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948, 1995.

[5] R. Storn and K. Price. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces, 1995.

[6] Xin-She Yang. *A New Metaheuristic Bat-Inspired Algorithm*, pages 65–74. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[7] M. M. al Rifaie and A. Aber. *Dispersive Flies Optimisation and Medical Imaging*, pages 183–203. Springer International Publishing, Cham, 2016.

[8] H. P. Borges, O. A. C. Cortes, and D. Vieira. An adaptive metaheuristic for unconstrained multimodal numerical optimization. In P. Korošec, N. Melab, and El-G. Talbi, editors, *Bioinspired Optimization Methods and Their Applications*, pages 26–37, Cham, 2018. Springer International Publishing.

[9] O. A. C. Cortes, A. Rau-Chaplin, and R. F. Lopes. A pso-based algorithm with local search for multimodal optimization without constraints. In *XXXVIII Conferencia Latinoamericana En Informatica (CLEI)*, pages 1–7, Oct 2012.

[10] E. Carvalho, O. A. C. Cortes, J. P. Costa, and D. Vieira. A parallel adaptive genetic algorithm for unconstrained multimodal numerical optimization. In *Simpósio Brasileiro de Automação Inteligente (SBAI)*, Oct 2017.