Um Algoritmo Híbrido Baseado em Fireworks e Evolução Diferencial para Otimização Numérica

Isaac Francis Curvelo Marques
Programa de Pós-Graduação em Engenharia da Computação e Sistemas (PECS)
Universidade Estadual do Maranhão (UEMA)
São Luis, Maranhão
isaacfrancis-cm@hotmail.com

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

ABSTRACT

This paper presents a cooperative hybrid algorithm based-on two metaheuristics: Fireworks and Differential Evolution. The main idea is to randomly choose which metaheuristic will be used on each iteration during the optimization process. We adopted the Rosenbrock function in our preliminary experiments. It is a challenging function to optimize because it is multimodal and non-separable when solving it in more than three dimensions. The initial investigations show that the hybrid algorithm tends to present better results than the canonical ones.

KEYWORDS

Meta-Heurísticas, Fireworks, Evolução Diferencial, Rosenbrock, Otimização

1 INTRODUÇÃO

Algoritmos híbridos tem recebido atenção na área devido aos bons resultados que eles podem alcançar, sendo que no pior caso eles tendem a apresentar resultados similares a aqueles apresentados pelos algoritmos canônicos. Um algoritmo híbrido junta características de duas ou mais meta-heurísticas para solucionar um problema. O desafio então é juntar as características de modo a não construir um algoritmo que seja desnecessariamente lento, em especial em aplicações cujas funções objetivo já são computacionalmente intensas por si só. Nesse sentido, a ideia deste trabalho é criar um algoritmo híbrido que alterne aleatoriamente entre a execução do algoritmo Fireworks [1] e de Evolução Diferencial [2] em cada iteração, criando um algoritmo que em essência é cooperativo e que compartilha a mesma população. A principal vantagem de usar esse algoritmo é que ambas meta-heurísticas colaboram na busca pela solução global.

Para testar o algoritmo utilizou-se a função Rosenbrock [3] que tem algumas características interessantes. A primeira delas é que ela é multimodal para dimensões maiores ou iguais a três [4]. A segunda é ser uma função não separável, ou seja, não é possível separar a função em equações menores, pois há dependência entre as variáveis, o que torna a função mais difícil de otimizar. Nesse contexto, a função Rosenbrock é considerada um desafio para muitos pesquisadores no campo da otimização numérica [5].

Assim, este artigo esta dividido da seguinte forma: a Seção 2 ilustra os conceitos básicos sobre otimização numárica; a Seção 3 mostra o funcionamento do algoritmo fireworks e como o algoritmo híbrido foi implementado; a Seção 4 apresenta um experimento inicial com o algoritmo híbrido e uma comparação com os algoritmos canônicos usados neste trabalho; finalmente a Seção 5 apresenta as conclusões e os trabalhos futuros.

2 OTIMIZAÇÃO NUMÉRICA

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 [6]. Basicamente, otimizar uma função numérica sem restrições é maximizar ou minimizar a função em questão [7].

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 $f(x*) > f(x)\ \forall\ x$. Nesse contexto, a função Rosenbrock constitui-se de um problema de minimização de acordo com a Equação 1, sendo seu mínimo igual a zero na posição $\{1, 1, \ldots, 1\}$.

$$f_1(x) = \sum_{i=1}^{n} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2] \quad (1)$$

3 FIREWORKS E O ALGORITMO HÍBRIDO

Quando fogos de artifícios são disparados, faíscas costumam preencher o espaço ao redor da explosão. Esse processo pode ser simulado e utilizado como um algoritmo de busca. A ideia é explodir fogos de artifícios em diversos locais até que as faíscas encontrem o ótimo global que se esta buscando. Esse processo pode ser visto na Figura 1, na qual o algoritmo do Fireworks é apresentado no formato de fluxograma.

O primeiro passo no algoritmo é escolher n locais para explodir os fogos de artifício. Depois da explosão, os locais das faíscas devem ser identificados e avaliados. Se o critério de parada for atingido o algoritmo é finalizado. Caso contrario, n novos locais, só que agora dentre as faíscas identificadas, são selecionados para novas explosões.

Um conceito importante quando as explosões dos fogos são realizadas é amplitude da explosão. Uma boa explosão tem amplitude pequena e as faíscas estão todas relativamente próximas ao local da explosão. Por outro lado, uma explosão ruim tem uma amplitude grande e as faíscas estão mais espalhadas pelo espaço. Levando isso em consideração, na busca global no inicio do algoritmo devem ser realizadas explosões ruins. Enquanto que no final, a busca local deve ser feita por explosões boas.

A quantidade de faíscas geradas em cada explosão é dada pela Equação 2, na qual m controla a quantidade total de faíscas dos n

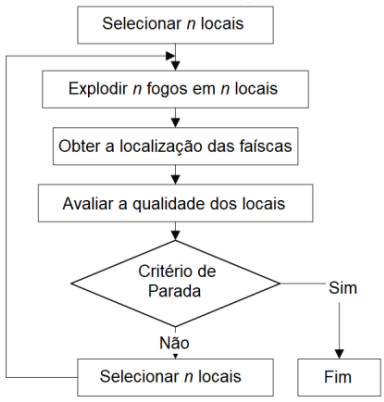

Figura 1: Algoritmo Fireworks no formato de fluxograma

fogos de artifício, y_{max} é o pior valor da função objetivo entre os n fogos, e ξ é uma constante bem pequena para evitar a divisão por zero.

$$s_i = m \times \frac{y_{max} - f(x_i) + \xi}{\sum_{i=1}^{n} (y_{max} - f(x_i)) + \xi} \tag{2}$$

Para evitar fogos de artifícios exagerados, são estipulados limites para s_i (Eq. 2) utilizando a Equação 3, na qual a e b são constantes definidas pelo usuário.

$$\hat{s}_i = \begin{cases} round(a \times m) & , se\ s_i < a\ .\ m \\ round(b \times m) & , se\ s_i > b\ .\ m\ e\ a < b < 1 \\ round(s_i) & , \text{caso contrário} \end{cases} \tag{3}$$

A amplitude para cada explosão é dada pela Equação 4, na qual $\hat{A}$ é a amplitude máxima, y_{min} é o melhor valor das n explosões e ξ novamente objetiva evitar a divisão por zero.

$$A = \hat{A} \times \frac{f(x_i) - y_{min} + \xi}{\sum_{i=1}^{n} (f(x_i) - y_{min}) + \xi} \tag{4}$$

Já as faíscas de cada explosão devem ser geradas em z dimensões através de $z = round(d \times rand(0,1))$, na qual d é a dimensão do problema (quantidade de variáveis) e $rand(0,1)$ é um valor distribuído uniformemente no intervalo [0,1]. As faíscas funcionam basicamente como um operador de mutação, similar ao que acontece nos Algoritmo Genéticos e na Evolução Diferencial. O Algoritmo 1 mostra como determinar a posição das faíscas que é basicamente adicionar h em cada dimensão escolhida. Lembrando que A é a amplitude da explosão. Outra forma de gerar faíscas é substituir o h por $h = Gaussian(1,1)$, que significa gerar uma número aleatório com distribuição Gaussiana com média 1 e desvio padrão igual a 1. Neste trabalho utiliza-se a geração gaussiana em M_g faíscas em cada iteração como sugerido no trabalho de Tan [8].

A última operação de acordo com a Figura 1 é escolher as n novas posições a partir das faíscas. Isso é feito da seguinte forma. A melhor localização $x*$ sempre será mantida para a próxima explosão. Dessa forma, devem ser escolhidas $n-1$ posições dentre as faíscas. Para isso calcula-se a distância entre x_i e os demais locais usando $R(x_i) = \sum_{j \in K} d(x_i, x_j) = \sum_{j \in K} ||x_i - x_j||$, na qual K é o conjunto de todas as localizações tanto dos fogos quanto das faíscas. Então

Algoritmo 1 - Geração de faíscas

$\tilde{x}_i = x_i$
$z = round(d \times rand(0,1))$
$h = A_i \times Rand(-1,1)$
para (Cada dimensão j escolhida em z de $\tilde{x}$) **faça**
  $x_j = x_j + h$
  VerificarLimites(x_j)
fim para

a probabilidade de se selecionar uma localização x_i é dada pela Equação 5, sendo que a distância pode ser calculada usando qualquer tipo, como por exemplo, distância Euclidiana, de Manhattan, dentre outras.

$$p(x_i) = \frac{R(x_i)}{\sum_{j \in K} R(x(j))} \tag{5}$$

Com relação ao algoritmo híbrido cooperativo a ideia é simplesmente escolher aleatoriamente, com uma distribuição uniforme, qual algoritmo executar em cada iteração. Para garantir que critério de parada seja obedecido pelo Fireworks e pelo algoritmo híbrido, a quantidade de faíscas é cortada aleatoriamente antes de chamar a função objetivo de modo a garantir 50 locais para a próxima iteração. Assim, caso a iteração atual seja de Fireworks e a próxima iteração seja de ED, garante-se uma população de 50 indivíduos. Caso a próxima iteração seja novamente de Fireworks seleciona-se os n indivíduos, sendo um a melhor solução e $n-1$ soluções aleatórias. Enquanto que no Fireworks, as $n-1$ melhores soluções são escolhidas com base na Equação 5.

Por questões de espaço não será mostrado o algoritmo de evolução diferencial que já é bem conhecido na literatura e pode ser entendido no trabalho de Storn [2].

4 EXPERIMENTO

O experimento foi executado em um computador i5 com 16 GB de RAM e Windows 10 de 64 bits e linguagem Python 3. Foram utilizadas 50.000 chamadas à função objetivo, população igual a 50 indivíduos na ED e 30 execuções. Os parâmetros utilizados em cada algoritmo estão descritos na Tabela 1.

Tabela 1: Parâmetros utilizados

Algoritmo	Parâmetros
Fireworks	$Mg = 5$, $n = 5$, $m = 50$, $a = 0.04$, $b = 0.8$ e $\hat{A} = 40$
ED	$p_c = 0.8$, $F = 0.6$
Híbrido	$Mg = 5$, $n = 5$, $m = 50$, $a = 0,04$, $b = 0,8$ e $\hat{A} = 40$, $p_c = 0.8$ e $F = 0.6$

A Tabela 2 mostra os resultados dos algoritmos, sendo apresentadas a melhor solução, a média e o desvio padrão. Embora na média os algoritmos apresentem resultados similares, a melhor solução é apresentada pelo algoritmo híbrido.

OS tempos de execução são apresentados na Figura 2, na qual pode-se observar que o algoritmo híbrido é mais lento que a ED canônica, porém, mais rápido que o Fireworks, que é um resultado esperado.

Tabela 2: Resultados - 50.000 chamadas à função objetivo

	Fireworks	ED	Híbrido
Menor	26,0107	19,8762	**0,3962**
Média	26,9453	**21,9669**	22,3487
Desvio Padrão	0,5532	0,9449	9,7568

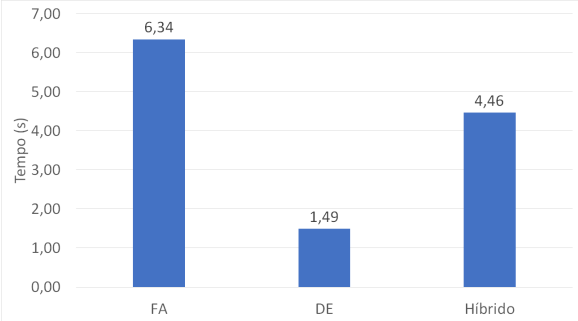

Figura 2: Tempo médio por execução em segundos

5 CONCLUSÕES

Neste trabalho apresentou-se um algoritmo cooperativo híbrido baseado nos algoritmos de Fireworks e Evolução Diferencial. Resultados preliminares indicaram que o algoritmo híbrido alcança o melhor resultado dentre as 30 execuções na otimização da função Rosenbrock.

Trabalhos futuros incluem: aumentar a quantidade de funções de benchmarks utilizadas; variação na forma de hibridização de modo a diminuir o desvio padrão do algoritmo; implementar uma forma adaptativa estocástica de escolher o algoritmo a ser executado em cada iteração; criar um sistema fuzzy para determinar qual algoritmo executar; e paralelizar a execução do algoritmo híbrido em aplicações do mundo real.

REFERÊNCIAS

[1] Ying Tan and Yuanchun Zhu. Fireworks algorithm for optimization. In Ying Tan, Yuhui Shi, and Kay Chen Tan, editors, *Advances in Swarm Intelligence*, pages 355–364, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

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

[3] H. H. Rosenbrock. An automatic method for finding the greatest or least value of a function. *Computer Journal*, 3(3):175–184, 1960.

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

[5] Peter A. N. Bosman and Dirk Thierens. *Numerical Optimization with Real-Valued Estimation-of-Distribution Algorithms*, pages 91–120. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[6] 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.

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

[8] Y. Tan and Y. Zhu. Fireworks algorithm for optimization. In Ying Tan, Yuhui Shi, and Kay Chen Tan, editors, *Advances in Swarm Intelligence*, pages 355–364, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.