

# MOTH-FLAME OPTIMIZATION

Bardia Nikbakhsh

bardian@USTMB.ac.ir








# Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm

Seyedali Mirjalili  

Show more 

 Add to Mendeley  Share  Cite

<https://doi.org/10.1016/j.knosys.2015.07.006> 

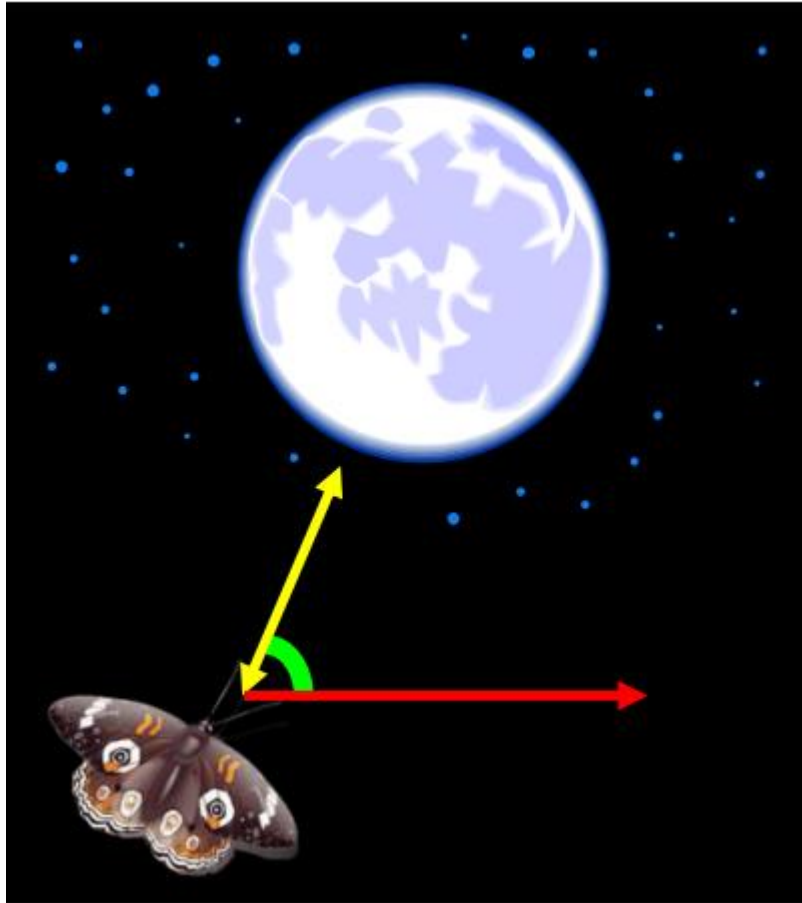
[Get rights and content](#) 

## Abstract

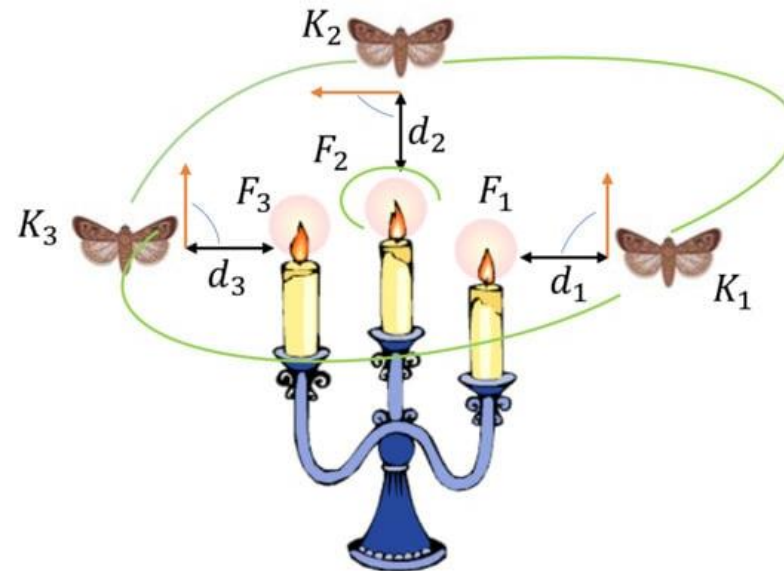
In this paper a novel nature-inspired optimization paradigm is proposed called Moth-Flame Optimization (MFO) algorithm. The main inspiration of this optimizer is the navigation method of moths in nature called transverse orientation. Moths fly in night by maintaining a fixed angle with respect to the moon, a very effective mechanism for travelling in a straight line for long distances. However, these fancy insects are trapped in a useless/deadly spiral path around artificial lights. This paper mathematically models this behaviour to perform optimization. The MFO algorithm is compared with other well-

■ الگوریتم شمع و پروانه یا MFO یکی از الگوریتم های بهینه سازی و فراابتکاری است که از رفتار پروانه ها در کنار شعله یا آتش روشی برای حل مسئله پیدا می کند. این الگوریتم در سال ۲۰۱۵ توسط سید علی میرجلیلی مطرح شد.

# TRANSVERSE ORIENTATION



■ الهام بخش اصلی این بهینه ساز روش ناوبری پروانه ها در طبیعت به نام جهت گیری عرضی یا **transverse orientation** است.



# MFO

■ در الگوریتم MFO، فرض بر این است که راه حل های کاندید پروانه ها هستند و متغیرهای مسئله موقعیت پروانه ها در فضا است. بنابراین، پروانه ها با تغییر بردارهای موقعیتی خود می توانند در فضای یک بعدی، دو بعدی یا سه بعدی پرواز کنند. از آنجا که الگوریتم MFO یک الگوریتم مبتنی بر جمعیت است، مجموعه پروانه در یک ماتریس ( $M$ ) نمایش داده می شوند.

$$M = \begin{bmatrix} m_{1,1} & m_{1,2} & \dots & \dots & m_{1,d} \\ m_{2,1} & m_{2,2} & \dots & \dots & m_{2,d} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ m_{n,1} & m_{n,2} & \dots & \dots & m_{n,d} \end{bmatrix}$$

# MFO

■ آرایه ای نیز برای تمامی پروانه ها برای ذخیره مقادیر تناسب (OM) وجود دارد. یکی دیگر از مؤلفه های اصلی در الگوریتم یک ماتریس شبیه به ماتریس پروانه ها است که ماتریس شعله یا آتش (F) است و یک آرایه نیز با نام OF برای ذخیره کردن مقدار تابع تناسب آن استفاده می شود.

- **F:** ماتریس شعله
- **M:** ماتریس پروانه
- **OF:** ماتریس مقادیر تناسب پروانه
- **OM:** ماتریس مقادیر تناسب پروانه

$$OM = \begin{bmatrix} OM_1 \\ OM_2 \\ \vdots \\ OM_n \end{bmatrix}$$

$$F = \begin{bmatrix} F_{1,1} & F_{1,2} & \dots & \dots & F_{1,d} \\ F_{2,1} & F_{2,2} & \dots & \dots & F_{2,d} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ F_{n,1} & m_{n,2} & \dots & \dots & F_{n,d} \end{bmatrix}$$

# MFO

- **M**: ماتریس پروانه
- **OM**: ماتریس مقادیر تناسب پروانه
- **F**: ماتریس شعله
- **OF**: ماتریس مقادیر تناسب پروانه

$$M = \begin{bmatrix} m_{1,1} & m_{1,2} & \cdots & \cdots & m_{1,d} \\ m_{2,1} & m_{2,2} & \cdots & \cdots & m_{2,d} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ m_{n,1} & m_{n,2} & \cdots & \cdots & m_{n,d} \end{bmatrix} \rightarrow OM = \begin{bmatrix} OM_1 \\ OM_2 \\ \vdots \\ OM_n \end{bmatrix}$$

$$F = \begin{bmatrix} F_{1,1} & F_{1,2} & \cdots & \cdots & F_{1,d} \\ F_{2,1} & F_{2,2} & \cdots & \cdots & F_{2,d} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ F_{n,1} & F_{n,2} & \cdots & \cdots & F_{n,d} \end{bmatrix} \rightarrow OF = \begin{bmatrix} OF_1 \\ OF_2 \\ \vdots \\ OF_n \end{bmatrix}$$

# HOW IT WORKS

1. در ابتدا، یک جمعیت از پروانه ها در فضای جستجو قرار می گیرند.
2. هر پروانه با سرعت و جهت متفاوتی به سمت شعله حرکت می کند.
3. پس از هر تکرار، پروانه ها با توجه به عملکردشان در تکرار قبلی، بروزرسانی می شوند.
4. این فرآیند تا زمانی که شرایط توقف الگوریتم محقق نشود، ادامه پیدا می کند.

# PARAMETERS

## ■ پارامتر $a$

■ پارامتر  $a$  یک عامل مقیاس‌گذاری است که بر میزان حرکت پروانه‌ها تأثیر می‌گذارد. مقدار  $a$  باید به گونه‌ای انتخاب شود که پروانه‌ها بتوانند به طور مؤثری در فضای جستجو حرکت کنند.

## ■ پارامتر $\beta$

■ پارامتر  $\beta$  یک عامل تصادفی است که بر میزان بی‌نظمی در حرکت پروانه‌ها تأثیر می‌گذارد. مقدار  $\beta$  باید به گونه‌ای انتخاب شود که پروانه‌ها بتوانند به طور مؤثری از بهینه‌های محلی فرار کنند.



# OTHER PARAMETERS

## ■ اندازه جمعیت

■ اندازه جمعیت تعداد پروانه‌های موجود در جمعیت را تعیین می‌کند. اندازه جمعیت باید به گونه‌ای انتخاب شود که الگوریتم بتواند بهینه‌های جهانی را پیدا کند.

## ■ تعداد تکرارها

■ تعداد تکرارها تعداد دفعاتی است که الگوریتم تکرار می‌شود. تعداد تکرارها باید به گونه‌ای انتخاب شود که الگوریتم بتواند بهینه‌های جهانی را پیدا کند

# PSEUDO-CODE

---

**Algorithm 1.** Pseudo-code for improved moth flame optimization (IMFO) algorithm.

---

*Initialization of location of moths in the search space*

**While**(iteration <= Maximum iteration)

*Update the number of flames by utilizing Equation (9)*

*OM = Fitness Function (Equation (8))*

**if** iteration = 1

*F = sort(M);*

*OF = sort (OM);*

**else**

*F = sort ( $K_{t-1}$ ,  $K_t$ );*

*OF = sort ( $K_{t-1}$ ,  $K_t$ );*

**end**

*for* i = 1: n

*for* j = 1: d

*Calculate D using Equation (12) with respect to the corresponding moth*

*Update K(i,j) using Equations (10), (11) and (13) with respect to the corresponding moth*

**end**

**end**

---

# IMPLEMENTATION

```
import random

def objective_function(x):
    return x[0]**2 - 2 * x[0] * x[1] + x[1]**2
```

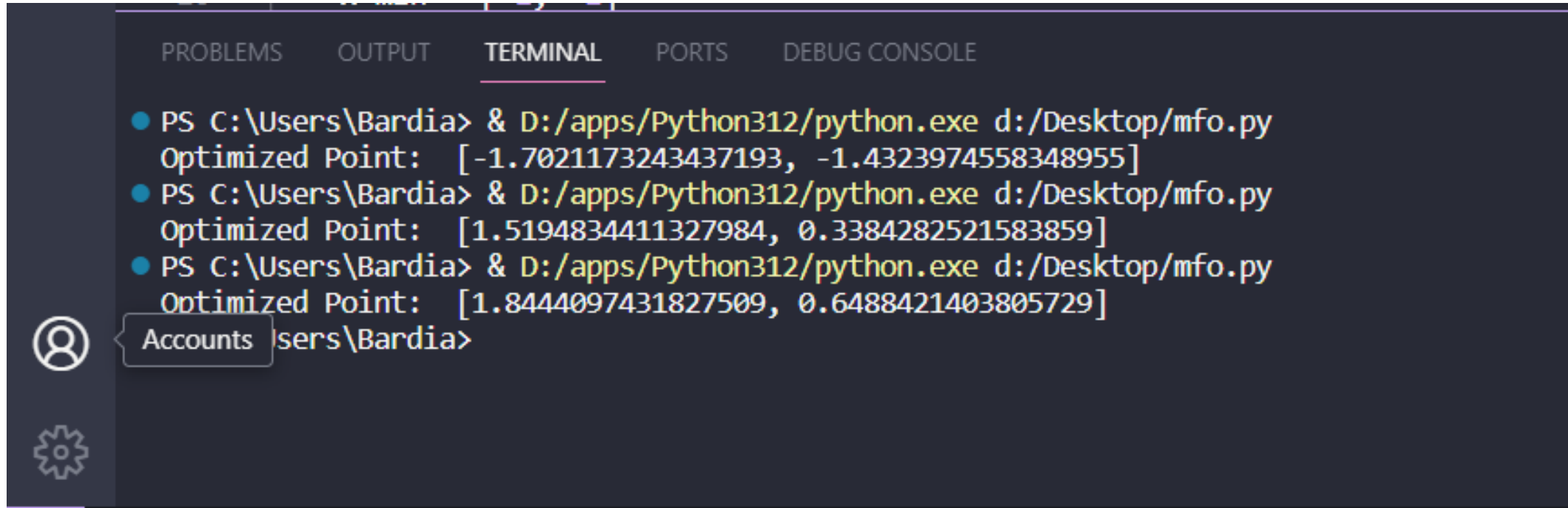
```
def moth_flame_optimization(x_min, x_max, max_iters, alpha, beta):  
    x = []  
    for i in range(len(x_min)):  
        x.append(random.uniform(x_min[i], x_max[i]))  
  
    for j in range(max_iters):  
        speed = alpha * random.uniform(0, 1)  
        direction = beta * random.uniform(-1, 1)  
  
        x_new = []  
        for i in range(len(x)):  
            x_new.append(x[i] + speed * direction)  
        score_current = objective_function(x)  
        score_new = objective_function(x_new)  
  
        if score_new < score_current:  
            x = x_new  
  
    return x
```

# IMPLEMENTATION

```
x_min = [-1, -1]
x_max = [1, 1]
max_iters = 1000
alpha = 0.5
beta = 0.7

result = moth_flame_optimization(x_min, x_max, max_iters, alpha, beta, gamma)
```

# RESULT



The screenshot shows a VS Code interface with a terminal window open. The terminal has tabs for PROBLEMS, OUTPUT, TERMINAL, PORTS, and DEBUG CONSOLE. The TERMINAL tab is active, displaying three lines of command execution and their results. On the left side of the terminal, there is a sidebar with a user icon and a gear icon. A tooltip labeled 'Accounts' is visible over the user icon, showing 'Users\Bardia'.

```
PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE
```

- PS C:\Users\Bardia> & D:/apps/Python312/python.exe d:/Desktop/mfo.py  
Optimized Point: [-1.7021173243437193, -1.4323974558348955]
- PS C:\Users\Bardia> & D:/apps/Python312/python.exe d:/Desktop/mfo.py  
Optimized Point: [1.5194834411327984, 0.3384282521583859]
- PS C:\Users\Bardia> & D:/apps/Python312/python.exe d:/Desktop/mfo.py  
Optimized Point: [1.8444097431827509, 0.6488421403805729]

Accounts Users\Bardia>

# ADVANTAGES

- سرعت
- تعمیم پذیری
- انعطاف پذیری

# DISADVANTAGES

■ احتمال گیر کردن در نقطه محلی بهینه



# USECASES

- استفاده از الگوریتم MFO برای طراحی مدارهای الکتریکی
- الگوریتم MFO را می توان برای طراحی مدارهای الکتریکی با بهینه سازی پارامترهای مدار مانند مقاومت، ظرفیت و اندوکتانس اجزای آن استفاده کرد.
- استفاده از الگوریتم MFO برای بهینه سازی طراحی سازه ها
- الگوریتم MFO را می توان برای بهینه سازی طراحی سازه هایی مانند ساختمان ها و پل ها استفاده کرد. این امر می تواند منجر به سازه هایی شود که قوی تر، سبک تر و بادوام تر هستند.
- استفاده از الگوریتم MFO برای برنامه ریزی تولید
- الگوریتم MFO را می توان برای برنامه ریزی تولید با بهینه سازی تخصیص منابع مانند مواد، نیروی کار و تجهیزات استفاده کرد.
- استفاده از الگوریتم MFO برای مدیریت منابع
- الگوریتم MFO را می توان برای مدیریت منابع مانند آب، انرژی و زمین استفاده کرد. این امر می تواند منجر به استفاده کارآمدتر و عادلانه تر از منابع شود.

**THE END**

