# 作業二

**R26094022 統計所 李權恩**

## Abstract

In this project, we use three meta-heuristic algorithms: differential evolution (DE), particle swarm optimization (PSO), and firefly algorithm (FA) to find the global optimum of the objective function (briefly introduced in the first section). In the numerical experiment, we noticed that FA takes more time.

## 1. Problem Statement

Suppose $\xi = \begin{pmatrix} x_1 & x_2 & x_3 \\ w_1 & w_2 & w_3 \end{pmatrix}$ with $\sum_{i=1}^{3} w_i = 1$ and $0 \le x_i \le 30 \ \forall i$, define a function $M(\xi, \theta^0) = \sum_i w_i f(x_i, \theta^0) f^T(x_i, \theta^0)$ .The goal is to $\max_{\xi} |M(\xi, \theta^0)|$ with the following settings：$\theta^0$ is a fixed numeric vector such that $\theta^0 = [0.05884, 4.298, 21.8]^T$ , and $f(x, \theta^0) = [f_1(x, \theta^0), f_2(x, \theta^0), f_3(x, \theta^0)]^T = [x\theta_3 \exp(-\theta_1 x), -x\theta_3 \exp(-\theta_2 x), \exp(-\theta_2 x) - \exp(-\theta_1 x)]^T$. In addition, we set $x_1 \le x_2 \le x_3$ to avoid multiple solutions of $\xi$.

## 2. Methodology

According to our purpose, we use three different meta-heuristic algorithms: differential evolution (DE), particle swarm optimization (PSO) and firefly algorithm (FA). These three algorithms are derivatives-free method. We will give a simple introduction to the algorithm in this section.

### 2.1. Differential Evolution

At the beginning in DE, we generate several random initialization vectors(population) in the search space. All vectors perturb each other in each iteration. Similar to genetic algorithm with each iteration, each vector is updated, using mutation, crossover, and selection to find the better solution.

Consider we are updating the current vector, three individual are randomly selected from population. The mutation is creating a donor vector by adding a weighted difference of two individual to the other; crossover is mix up the current vector and donor vector according to some rules to generate the experimental vector; select is choosing the vector from current vector and experimental vector which have better solution.

Algorithm：

1. Randomly generate $N$ initial population. $x_i^0 = (x_{i1}^0, x_{i2}^0, \dots, x_{id}^0)^T, i = 1, \dots, N$, and set the weight $F \in (0,2]$ and crossover probability $C_r \in (0,1)$.
2. Let $x_i^t$ be the current vector with each iteration：
   Mutation：$v_i^{t+1}$ represent the donor vector in $t+1$ times iterations and $x_p^t, x_q^t, x_r^t$ are randomly select from population.
   $$v_i^{t+1} = x_p^t + F\left(x_q^t - x_r^t\right)$$
   Crossover：mix up the current vector and donor vector according to the rules.

$$u_{j,i}^{t+1} = \begin{cases} v_{j,i}^{t+1} , & if \ \ r_i^t \le C_r, or \ \ j = J_r \\ x_{j,i}^{t+1} , & otherwise \end{cases}, r_i^t \in (0,1), J_r \in \{1, \dots, d\}$$

Selection：Consider the maximization problem.

$$x_i^{t+1} = \begin{cases} u_i^{t+1}, if \ \ f(u_i^{t+1}) \ge f(x_i^t) \\ x_i^t \quad , otherwise \end{cases}$$

## 2.2. Particle Swarm Optimization

In the particle swarm optimization algorithm, each particle has its own speed in the search space, and the search strategy is adjusted according to its past experience and group behavior. Similar to the mutation mechanism of genetic algorithm, but the difference from typical mutation is that it is not completely random. Many research results show that this method can quickly find the best solution in the problem space.

Algorithm：

1. Randomly generate $N$ initial population. $x_i^0 = (x_{i1}^0, x_{i2}^0, \dots, x_{id}^0)^T, i = 1, \dots, N$, set the weights $\alpha, \beta > 0$, set inertia function $\theta(t)$, set initial velocity $v_i^0 = 0$ and find $g^*$ from $max\{f(x_1^0), f(x_2^0), \dots, f(x_N^0)\}$ (at $t = 0$).
2. With each iteration：
   (1) Generate new velocity：
   $$v_i^{t+1} = \theta(t) v_i^t + \alpha \epsilon_1 \odot [g^* - x_i^t] + \beta \epsilon_2 \odot [x_i^{*(t)} - x_i^t]$$
   Where $\epsilon_1, \epsilon_2$ are random vector between 0 and 1, $g^*$ represent current global best and $x_i^{*(t)}$ represent the current best for particle $i$.
   (2) Generate new location：
   $$x_i^{t+1} = x_i^t + v_i^{t+1}$$
   (3) Update current best for each particle.
   (4) Update current global best.

## 2.3. Firefly Algorithm

In the FA, the idea is come from the blinking behavior of fireflies. The main purpose of the flash of fireflies is to act as a signal system to attract other fireflies. In order to idealized the characteristics of fireflies there have three assumptions:

1. All fireflies are genderless. Firefly will be attracted to other fireflies.
2. Attraction is directly proportional to the brightness of the light. For any two fireflies, the less bright one would move towards the brighter one. The attractiveness is proportional to the brightness and thus they both decrease as their distance increases.
3. The brightness of a firefly is affected or determined by the landscape of the objective function. As a result, for a maximization problem, the brightness can simply be proportional to the value of the objective function.

Algorithm：

1. Randomly generate $N$ initial population. $x_i^0 = (x_{i1}^0, x_{i2}^0, \dots, x_{id}^0)^T, i = 1, \dots, N$, set light absorption coefficient $\gamma$, set the attractiveness $\beta_0$ (when distance equal to zero) and let light intensity for each individual determined by objective function.

2. With each iteration:
   For every firefly $i$ and for every firefly $j$ without $i$ :
   (1) Move firefly $i$ to towards $j$ if light intensity for firefly $i$ if less then firefly $j$.

$$x_i^{t+1} = x_i^t + \beta_0 e^{-\gamma r_{ij}^2}\left(x_j^t - x_i^t\right) + \alpha\epsilon_i^t$$

   Where $\beta_0 e^{-\gamma r_{ij}^2}$ is the attractiveness function of distance $r_{ij}$ and $\epsilon$ is random vector(Gaussian or uniform).
   (2) Vary attractiveness with distance $r$ via $exp(-\gamma r^2)$.
   (3) Evaluate new solution and update light intensity.
   (4) Rank the fireflies and find the current global best $g^*$.

# 3. Numerical result

| **method** | **Hyper-parameter** | **value** |
|---|---|---|
| DE | Population size | 100 |
| | $F$(weight) | 0.6 |
| | Iteration number | 50 |
| | | |
| PSO | Population size | 50 |
| | $\alpha$ | 2 |
| | $\beta$ | 2 |
| | $\theta$ | 0.7 |
| | Iteration number | 50 |
| | | |
| FA | Population size | 30 |
| | $\alpha$ | 0.1 |
| | $\beta$ | 2 |
| | $\gamma$ | 0.1 |
| | $\theta$ | 0.99 |