Filomat 34:15 (2020), 5173–5185
https://doi.org/10.2298/FIL2015173L

Published by Faculty of Sciences and Mathematics,
University of Niš, Serbia
Available at: http://www.pmf.ni.ac.rs/filomat

Population Based Optimization via Differential Evolution and Adaptive Fractional Gradient Descent

Zijian Liu[a], Chunbo Luo[a,c], Peng Ren[b], Tingwei Wang[b], Geyong Min[c]

[a] *University of Electronic Science and Technology of China*
[b] *China University of Petroleum (East China)*
[c] *University of Exeter, UK*

Abstract. We propose a differential evolution algorithm based on adaptive fractional gradient descent (DE-FGD) to address the defects of existing bio-inspired algorithms, such as slow convergence speed and local optimum. The crossover and selection processes of the differential evolution algorithm are discarded and the adaptive fractional gradients are adopted to enhance the global searching capability. For the benchmark functions, our proposed algorithm Specifically, our method has higher searching accuracy than several state of the art bio-inspired algorithms. Furthermore, we apply our method to specific tasks – parameters estimation of system response functions and approximate value functions. Experiment results validate that our proposed algorithm produces accurate estimations and improves searching efficiency.

Introduction

Searching global optimum is a universal and challenging task to optimization methods [1]. Conventional gradients based search algorithms [2] were created to solve the convex optimization problem [3]. And they are widely applied to machine learning [4], pattern recognition [5] and other optimization tasks [6]. However, they are not applicable to non-convex, non-differentiable or discontinuous [7] problems. To solve these complex problems, bio-inspired algorithms (BIAs) [8] have been put forward.

Biological population consists of a group of individuals, and each one shows rather low-level competence and ad-hoc behaviors. Once the individuals in the group communicate with others, the group behaves intelligently and systemically. Inspired by intelligent and efficient biological mechanisms, many scientists present several BIAs. Kennedy et al. proposed a particle swarm optimization (PSO) [9] method to deal with the optimization problem. Inspired by the mechanism of ants for finding the shortest hunting path, Dorigo et al. proposed an ant colony system algorithm (ACS) [10] for solving the traveling salesman problem. Although the existing methods have achieved notable performance on optimization tasks, there is still slow convergence speed and easy to fall into local optimum [11].

2010 *Mathematics Subject Classification*. Primary 90C59
Keywords. Bio-inpsired algorithm, fractional gradients, approximate value function
Received: 02 November 2018; Accepted: 22 March 2019
Communicated by Shuai Li
Research supported partly by the University of Electronic Science and Technology of China 985 Project Fund (Project No. A10985xxx132), the National Natural Science Foundation of China (Project No. 61871096), and partly by the Qingdao Applied Fundamental Research (Project No. 16-5-1-11-jch)
Email addresses: liu6zijian@std.uestc.edu.cn (Zijian Liu), c.luo@uestc.edu.cn (Chunbo Luo)

To overcome these defects, we present a differential evolution algorithm based on adaptive fractional gradient descents, which shows satisfactory global search capacity than conventional gradient descent and other three state of the art methods – Particle Swarm Optimization (PSO) [9], Gravitational Search Algorithm (GSA) [12], Differential Evolution (DE) [13]. In addition, our method is applied to parameter estimation [14] of system response function and approximate function [15], to predict the output value and improve the efficiency of the robot seeking for the target.

We come up with an adaptive fractional gradient operator for the first time, which guarantees the solution to avoid falling into local optimum of fractional order. Moreover, we assign a novel rule to avoid initial position overlaps [16, 17] with fractional extreme points and calculated point.

This article is organized as follows. Section 1 presents a novel optimization method. Section 2 validates the performance of our method with different orders, and compare it with three state of the art methods – Particle Swarm Optimization, Differential Evolution Algorithm, Gravitational Search Algorithm. Section 3 applies proposed method into applications.

1. A Differential Evolution Algorithm based on Adaptive Fractional Gradient Descent

The task of optimization method is to find an optima (minimum or maximum) from feasible solutions space $\mathbf{x} \in \mathbf{X}$ for a certain problem, with respect to environmental condition function $f(\mathbf{x})$: $\mathbb{R}^D \to \mathbb{R}$. The usual aim is to obtain the optimal position $\mathbf{x}_\star$ that minimizes or maximizes the environmental condition function. This section proposes a novel evolution algorithm which combines advantages of DE and FGD methods. Adaptive fractional gradient operator ensures satisfactory global search competence, and Differential Evolution guarantees efficient convergence speed.

1.1. Differential Evolution

Differential evolution (DE) [13] is a method that optimizes a problem by iteratively improving a candidate solution. As one of the BIAs, DE is used for multi-dimensional functions but do not require for the optimization problem to be differentiable. And the computation processes are given as follows

1. **Step one: Initialization**
   All individuals $\mathbf{X}^0 = \{\mathbf{x}_1^0, \mathbf{x}_2^0, \cdots, \mathbf{x}_N^0\}$ are randomly generated in the domain
   $$\mathbf{X}^0 = (B_U - B_L)R(N, D) + B_L \tag{1}$$
   where B_U and B_L are the upper and lower boundary of the searching space. N is the number of population, and D is the dimension of each individual. The function $R(N, D)$ generates an $N \times D$ matrix, whose elements follow uniform distribution.
2. **Step two: Mutation**
   For an individual k at step t, if the environmental condition $f(\mathbf{x}_k^t)$ is the smallest one among all the environmental conditions $f(\mathbf{X}^t)$, we set the position of individual as the best position k at step t. All individuals are aggregated to the current best position
   $$\mathbf{v}_i^t = \mathbf{x}_i^t - \lambda(\mathbf{x}_i^t - \mathbf{x}_\star^t) \tag{2}$$
   $\lambda \in (0, 1)$ is the searching parameter, and $\mathbf{x}_i^t$ is the position of individual i at step t. Therefore, the generated vector $\mathbf{v}_i^t$ is an admixture of individuals $(\mathbf{x}_i^t, \mathbf{x}_\star^t)$.
3. **Step three: Selection**
   If the environmental condition of the generated position $\mathbf{v}_i^t$ is better than the original one, the individual i will move to the new position at step t+1 ($\mathbf{x}_i^{t+1} = \mathbf{v}_i^t$). Otherwise, it will remain on where it is ($\mathbf{x}_i^{t+1} = \mathbf{x}_i^t$). The new position of individual i at step $t + 1$ is
   $$\mathbf{x}_i^{t+1} = \begin{cases} \mathbf{v}_i^t, & f(\mathbf{v}_i^t) < f(\mathbf{x}_i^t) \\ \mathbf{x}_i^t, & f(\mathbf{v}_i^t) \geq f(\mathbf{x}_i^t) \end{cases} \tag{3}$$
   The optimal position is obtained by repeating the mutation and selection steps. However, DE models usually converge to the local optimum and the convergence speed is slow.

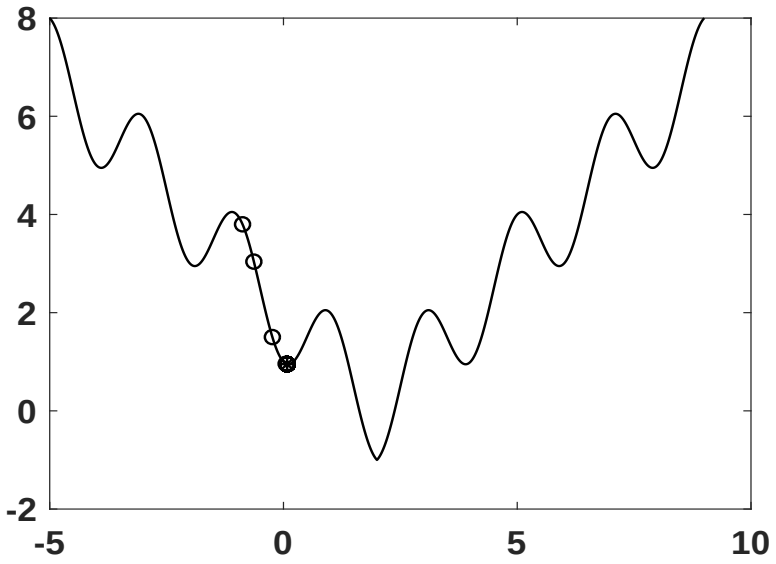


Figure 1: The computing result of GD method falls into the local optima

1.2. Gradient Descent

Different from DE method, gradient descent (GD) approach is a traditional searching algorithm, which ensures that GD usually finds a determined solution under the same initial conditions. At each step, it will descend along the steepest direction which is the first order gradient

$$\nabla f(x) = \frac{f(x) - f(x - \Delta x)}{\Delta x} \tag{4}$$

where ∇ denotes the first order gradient operator. And the iteration formula of GD is given as follows

$$\mathbf{x}^{t+1} = \mathbf{x}^t - \mu \nabla f(\mathbf{x}^t) \tag{5}$$

where μ is the learning rate. With step t increasing, candidate solution will follow along the steepest direction until the derivative is equal to zero.

The GD algorithm has been applied to numerous optimization tasks for its effectiveness and efficiency. However, the algorithm is easy to fall into local optimal solution (shown in Figure 1). In addition, the searching speed slows down when it gets close to the minimum point, because the gradient becomes small.

1.3. Fractional Gradient Descent

The fractional gradient descent (FGD) method is the generalization of GD method. Moreover, we assign a novel rule to avoid initial position overlaps with fractional extreme points and calculated point.

We generalize the Cauchy formula for repeated integration from natural number to real number and utilize the Gamma function to replace the discrete nature of the factorial function. Therefore, the formula of fractional integration is given as follows

$$(J^n f)(x) = \frac{1}{(n-1)!} \int_0^x (x - t)^{n-1} f(t) dt \tag{6}$$

$$(J^v f)(x) = \frac{1}{\Gamma(v)} \int_0^x (x - t)^{v-1} f(t) dt \tag{7}$$

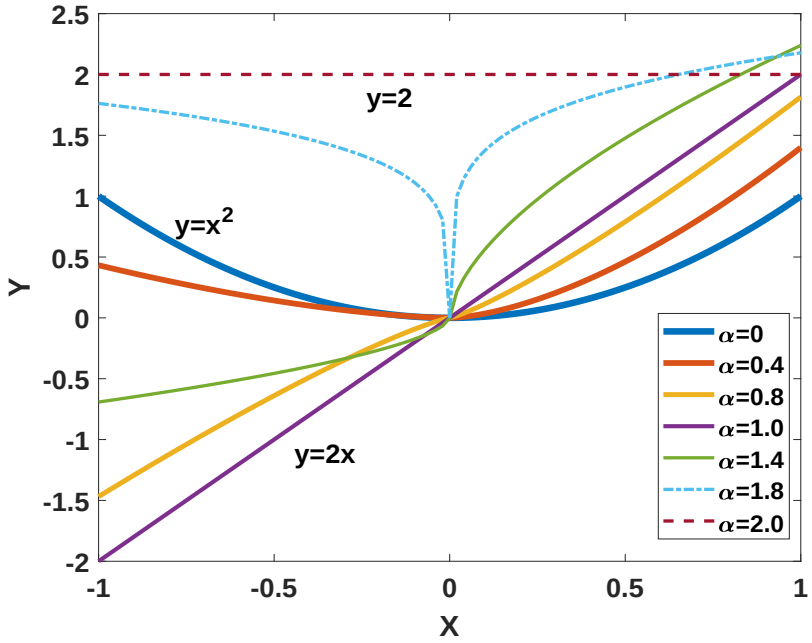


Figure 2: Fractional gradient of function $f(x) = x^2$ with different orders

where n is an integer, and ν is a real number. The gamma function is denoted as $\Gamma(\nu) = \int_0^\infty (x-t)^{\nu-1} f(t)dt$.

Owing to the reversibility of differentiation and integration, if $\nu = 1 - \alpha$, the formula of fractional difference is rewritten as follows

$$(J^{1-\alpha} f)(x) = \frac{1}{\Gamma(1-\alpha)} \int_0^x (x-t)^{-\alpha} f(t)dt \tag{8}$$

$$(D^{\alpha-1} f)(x) = \frac{1}{\Gamma(1-\alpha)} \int_0^x \frac{f(t)}{(x-t)^{\alpha}} dt \tag{9}$$

$$(D^{\alpha} f)(x) = \frac{1}{\Gamma(1-\alpha)} \frac{d}{dx} \int_0^x \frac{f(t)}{(x-t)^{\alpha}} dt \tag{10}$$

In addition, the Grünwald-Letnikov fractional discrete difference of order α is defined as follows

$$(D^{\alpha} f)(x) = \frac{1}{h^{\alpha}} \sum_{j=0}^{[\|\frac{x-a}{h}\|]} (-1)^j \frac{\Gamma(\alpha+1) f(x-jh)}{\Gamma(\alpha+j-1)\Gamma(j+1)} \tag{11}$$

where h denotes the step of calculation, $[a, x]$ is the duration of $f(x)$, and a is the initial point. $[A]$ means the maximum integer less than A. We use the following symbol to simplify the expression

$$\omega_j^{\alpha} = \begin{cases} 1, & j = 0 \\ \left(1 - \frac{\alpha+1}{j}\right) \omega_{j-1}^{\alpha}, & j \geq 1 \end{cases} \tag{12}$$

Then the iterative formula of fractional discrete difference is simplified as follows

$$(D^{\alpha} f)(x) = \frac{1}{h^{\alpha}} \sum_{j=0}^{[\|\frac{x-a}{h}\|]} \omega_j^{\alpha} f(x-jh) \tag{13}$$

Therefore, the formula of GD approach is generalized as the real order

$$\mathbf{x}^{t+1} = \mathbf{x}^t - \mu \mathrm{D}^{\alpha} f(\mathbf{x}^t) \tag{14}$$

With step t increasing, solution will descend along the fractional gradient until the derivative of fractional order is equal to zero. Furthermore, GD method is a special case of the FGD (fractional order $\alpha = 1$). Once the method is determined, a knotty issue need to deal with the sensitivity to the initial point a at each iteration. In general, a can be specified as an arbitrary value. However, when the initial point a overlaps with fractional extreme points or calculated points, the fractional derivative will become undefined and discontinuous. Therefore, a must be different from fractional extreme points or calculated points. Hence, we assign a novel rule to determine the boundary value as follows

$$a = (B_U - B_L)\,\mathrm{sgn}[(\mathbf{X}^0 - B_L) - (B_U - \mathbf{X}^0)] + B_L \tag{15}$$

where sgn[·] is a sign function. Therefore, when $(\mathbf{X}^0 - B_L) > (B_U - \mathbf{X}^0)$, the function value equals one. Otherwise, it becomes zero.

1.4. Overall Optimization

We propose a differential evolution algorithm based on adaptive fractional gradient descent (DE-FGD). In particular, an adaptive fractional gradient operator is presented for the first time. And it ensures more stable global search competence, which is given as follows

$$\alpha_i^{t+1} = \begin{cases} \alpha_i^t + \frac{t}{M}(1 - \alpha_i^t), & \|\mathbf{x}_i^{t+1} - \mathbf{x}_i^t\| < \varepsilon \\ \alpha_i^t, & otherwise \end{cases} \tag{16}$$

where M is the maximum iterative steps, and ε is the threshold which is small (we set ε to 0.001). Then our proposed method is presented as follows

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t - \lambda(\mathbf{x}_i^t - \mathbf{x}_\star^t) - \mu \mathrm{D}^{\alpha_i^t} f(\mathbf{x}_i^t) \tag{17}$$

where $\lambda(\mathbf{x}_i^t - \mathbf{x}_\star^t)$ is called DE part, and $\mu \mathrm{D}^{\alpha_i^t} f(\mathbf{x}_i^t)$ is called FGD part.

In this scenario, we combine gradient descent method with evolution algorithm, and remove the comparison process of DE part and update the fractional order from a constant α to a variable α_i^t of FGD part. In addition, the order α_i^t for different individual i is different. For one thing, each individual explores environment along their own fractional gradient $\mathrm{D}^{\alpha_i^t} f(\mathbf{x}_i^t)$. For another, they tend to aggregate the current best position $\mathbf{x}_\star^t$. For a specific instance, we assume that there are 9 agents, and the fifth agent is the current best $\mathbf{x}_\star^t$. Each agent explores along their own fractional gradient that is called FGD. At the same time, except the fifth agent, all agents tend to aggregate the current best agent, which is called DE. Therefore they follow the combination directions, which are called DE-FGD. And the evolution process of all agents searching global optimum at one step is shown in Figure 3 and DE-FGD method is demonstrated in Algorithm 1.

When the searching procedure falls into the local optimal solution, where $\|\mathbf{x}_i^{t+1} - \mathbf{x}_i^t\| < \varepsilon$ and $\mathrm{D}^{\alpha_i^t} f(\mathbf{x}_i^t) = 0$, we update the fractional order. Hence, $\mathrm{D}^{\alpha_i^{t+1}} f(\mathbf{x}_i^t) \neq \mathrm{D}^{\alpha_i^t} f(\mathbf{x}_i^t)$. Namely, $\mathrm{D}^{\alpha_i^{t+1}} f(\mathbf{x}_i^t) \neq 0$, and then it steps out the local optimal point. In the end, $\alpha_i^M = 1$, it will stop at the point where $\mathrm{D}^{\alpha_i^M} f(\mathbf{x}_i^t) = \nabla f(\mathbf{x}_i^t) = 0$. In addition, when $\lambda = 0$, Eq. 17 degrades into the adaptive fractional gradient descent method. When $\mu = 0$, the equation is converted to the differential evolutionary approach. Therefore, we regard FGD and DE methods as two special cases of our proposed DE-FGD algorithm.

2. Experiments

In this section, we firstly investigate the impact of different initial fractional orders to our proposed method. We make an empirical comparison with other state-of-the-art evolution algorithms, such as PSO, DE and GSA. All the experiments are conducted on five benchmark functions. Specifically, we test the convergence stability of all the models, visualizes the searching paths and evaluate their performance in terms of the optimal value they achieve.

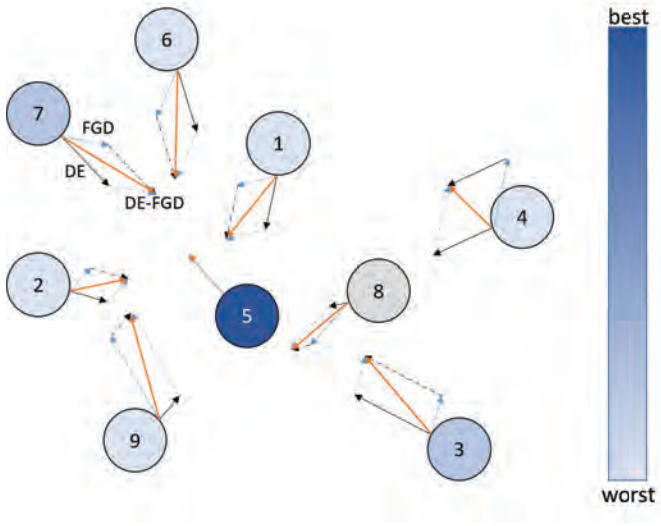


Figure 3: The evolution process of multiple agents searching global optimum at one step (taking the seventh agent for example, each one explore along the FGD direction and tends to aggregate the current best along the DE direction. The direction of vector composition is DE-FGD)

Algorithm 1: Differential Evolutionary Algorithm Based on Adaptive Fractional Gradient Descent

Input: Population size N; Dimension D; Maximum iteration steps Max_it
Output: Optimal solution $\mathbf{x}_\star$

Function
  Generate initial positions, $\mathbf{X}^0$;
  Define objective function, $f(\mathbf{x})$;
  Set an extremely minimum value container $f_\star$ (initialize to infinity);
  Determine initial fractional order α_j^0, $j = (1, 2, \cdots, N)$;
  while $t < Max\_it$ **do**
    Search for the minimum value of current positions, $\min\{f(\mathbf{X}^t)\}$ and $\arg\min_{\mathbf{x}_j^t}\{f(\mathbf{X}^t)\}$;
    if $\min\{f(\mathbf{X}^t)\} < f_\star$ **then**
      Update extremely minimum value container, $f_\star = \min\{f(\mathbf{X}^t)\}$;
      Update extremely solution container, $\mathbf{x}_\star = \arg\min_{\mathbf{x}_j^t}\{f(\mathbf{X}^t)\}$;
    end
    for $i = 1$ *to* N **do**
      Searh optimal solution iteratively, $\mathbf{x}_i^{t+1} = \mathbf{x}_i^t - \lambda(\mathbf{x}_i^t - \mathbf{x}_\star) - \mu D_{\mathbf{x}}^{\alpha_i^t} f(\mathbf{x}_i^t)$;
      if $\|\mathbf{x}_i^{t+1} - \mathbf{x}_i^t\| < \varepsilon$ **then**
        Update fractional order, $\alpha_i^{t+1} = \alpha_i^t + \frac{t}{Max\_it}(1 - \alpha_i^t)$;
      else
        Keep the order, $\alpha_i^{t+1} = \alpha_i^t$;
      end
    end
    $t = t + 1$;
  end
EndFun;

2.1. Performance evaluation of DE-FGD algorithm with different initial orders

In comparison with high risk of falling into local optimum of integral gradient methods, which is obtained by calculating the difference between current and adjacent value, the fractional gradient that is

gained by computing weighted values in the whole domain shows satisfactory global characteristic for searching. However, for different initial orders, the global searching performance of the algorithm will be different. Therefore it is particularly important to choose appropriate preliminary order for solving specific problems.

Here we investigate the influence of different initial fractional orders to our proposed method, and the experiments are conducted on a specific binary scalar value function, which contains several local optimal points. Thus finding the global optimum is a difficult task. And the binary scalar value function is represented as follows

$$f(x, y) = (x-1)^2 - \cos(2\pi x) + (y-1)^2 - \cos(2\pi y) \tag{18}$$

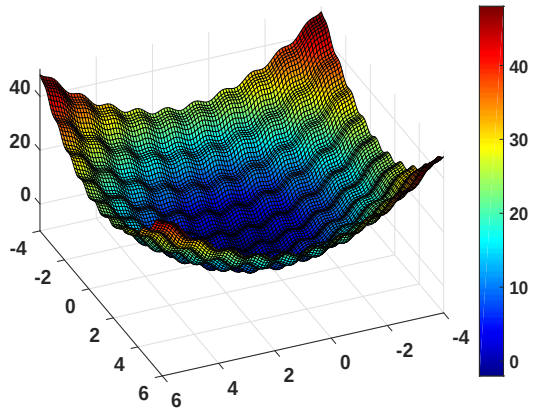


Figure 4: 3D image of function $f(x, y) = (x-1)^2 - \cos(2\pi x) - \cos(2\pi y)$

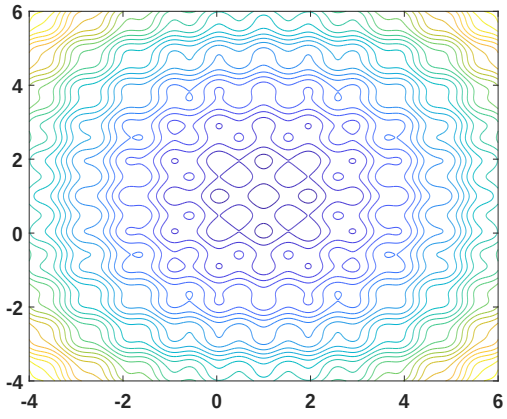


Figure 5: Contour image of function $f(x, y) = (x-1)^2 - \cos(2\pi x) - \cos(2\pi y)$

The domain of variable (x, y) is $[-4, 6]^2$. There are several local optimal points, and the only global optimal point is $(1, 1)$. The function is visualized as Figure 4 and Figure 5. Figure 4 is the 3D image of the function Eq.18, and Figure 5 shows the contour of the function Eq.18 from the top view. The darker color means the smaller value. We use the proposed method with different initial orders to search the global minimum. In this scenario, we just compare the searching performance of different initial orders. Considering the generality of the problem, random initialization will disturb the comparison. Therefore, we cut off the population based part (DE part), casually choose a fixed position with a single agent. The calculated results are shown in Figure 6 and Figure 7.

Figure 6 demonstrates that the searching paths of DE-FGD method with different initial orders are different. All of them are initialized from the same position $(-3, 4)$. All agents descent along the direction of their own fractional gradient, and these fractional orders tend to the first order. The obtained optimal solution will converge to the optimal point where the first derivative is equal to zero.

Figure 7 confirms that our proposed method is able to obtain the optimal solution more precisely with the order $\alpha = 1.6$ than other orders. And the obtained result with fractional order ($\alpha = 0.6, 0.8, 1.6$) is more accurate than the outcome found with gradient descent algorithm ($\alpha = 1$). Especially, when the order α is equal to 1.6, our proposed method will find the global optimal solution. When the order is less than 1.0, the computed result is more stable.

2.2. *Performance comparison of DE-FGD algorithm with other mainstream optimization methods*

We compare DE-FGD with other state-of-the-art optimization methods, including PSO, DE and GSA [12]. The experiments are conducted on five benchmark functions which are presented in Table 1, to verify

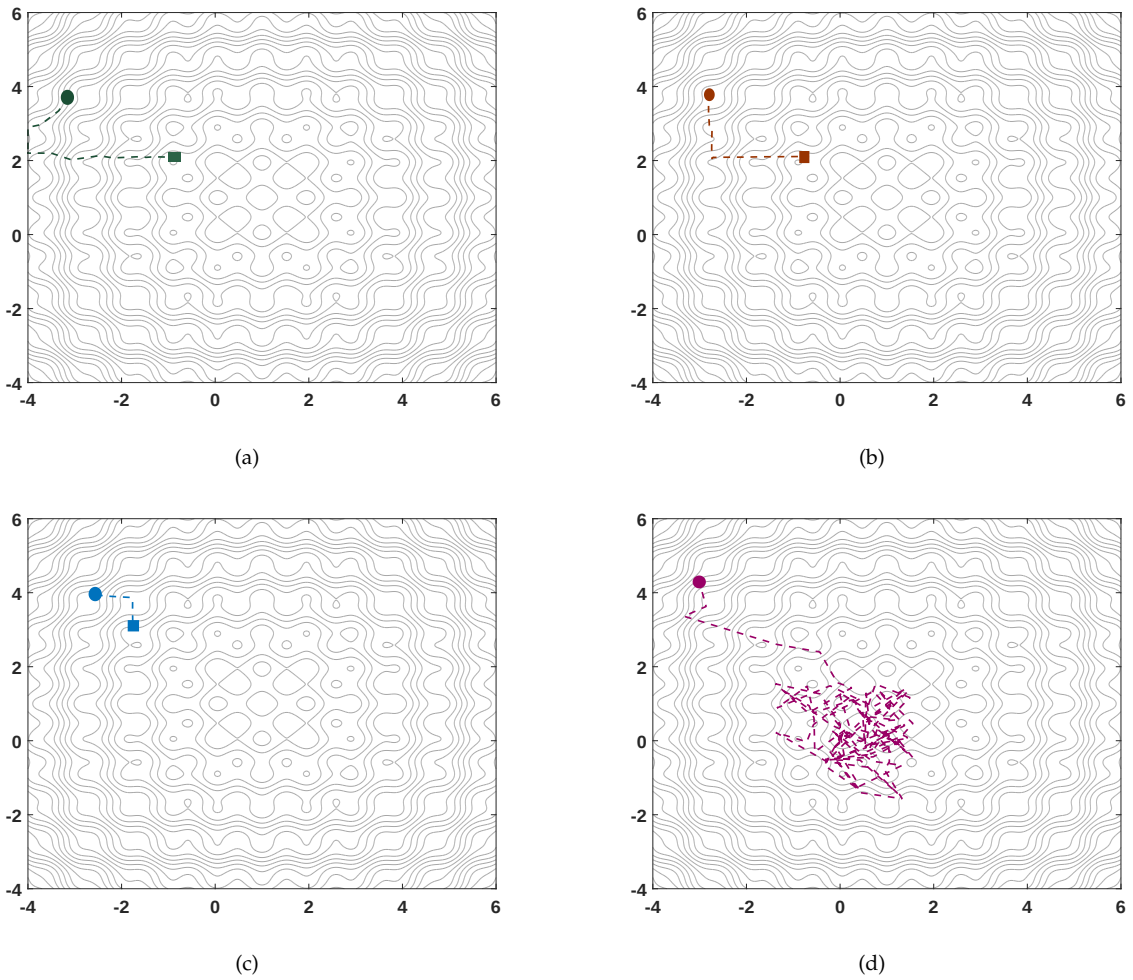


Figure 6: The minimum search trajectory on the function $f(x, y) = (x-1)^2 + (y-1)^2 - \cos(2\pi x) - \cos(2\pi y)$ utilizing DE-FGD method with different initial orders ($\alpha = 0.6$ (a), 0.8 (b), 1.0 (c), 1.6 (d))

the efficiency and effectiveness of our proposed method. We set the population size to 10, and the maximum iteration to 60. Table 2 shows the experimental results, which illustrates that proposed DE-FGD method is able to reach the global minimum more precisely than the other three methods on functions F_1, F_3, F_4, F_5. Especially, for function F_4, the other three optimizing methods converge to the local optimal solution, but our proposed method finds the global optimum.

3. Applications

In this section, we apply DE-FGD method to specific applications: utilizing DE-FGD algorithm to estimate parameters of the system response function, so as to realize the prediction of the output data. For a specific reinforcement learning model - a robot seeking for a target in the maze, parameters of an approximate function are estimated by our algorithm, to improve the efficiency of robot searching for the target.

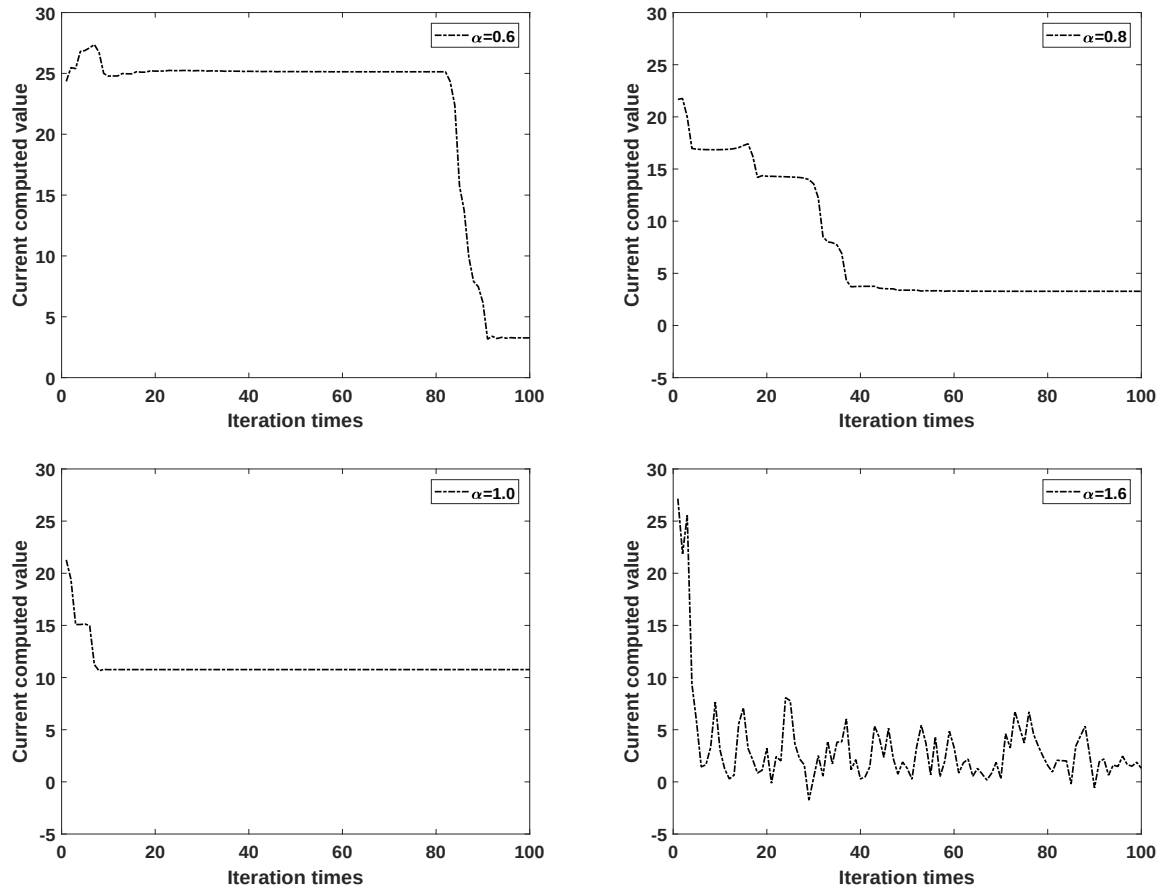


Figure 7: The minimum search iteration result on the function $f(x, y) = (x-1)^2 + (y-1)^2 - \cos(2\pi x) - \cos(2\pi y)$ utilizing DE-FGD method with different initial orders

Table 1: List of benchmark functions

Benchmarks	Domain	Minimum value
$F_1(X) = \sum_{i=1}^{D} x_i^2$	$[-100, 100]^D$	0
$F_2(X) = \sum_{i=1}^{D-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$	$[-30, 30]^2$	0
$F_3(X) = \sum_{i=1}^{D} \left[\sin(x_i)\sin(\frac{ix_i^2}{\pi})\right]^{20}$	$[-4, 4]^D$	-1.8013
$F_4(X) = 0.5 + \frac{\sin^2\sqrt{x_1^2+x_2^2}-0.5}{\left[1+0.001(x_1^2+x_2^2)\right]^2}$	$[-4, 4]^2$	0
$F_5(X) = \sum_{i=1}^{D} (x_i - 1)^2 - \cos(2\pi x_i)$	$[-4, 6]^D$	-2

3.1. Parameter estimation of system response function

In general, for an LTI (linear time invariant) SISO (Single-Input and Single-Output) system [20], it is usually modelled as follows

$$\sum_{j=0}^{n} a_{n-j} y(k-j) = \sum_{i=0}^{m} b_{m-i} x(k-i) \tag{19}$$

Table 2: Comparison of DE-FGD algorithm with three mainstream methods in benchmark functions

Best value	F_1	F_2	F_3	F_4	F_5
PSO	0.0242	**0.1158**	-1.7987	0.0097	-1.9993
DE	0.1161	0.2236	-1.7817	0.0097	-1.0466
GSA	1.2028	3.7705	-1.7480	0.0097	-1.9974
DE-FGD	**0.0099**	0.1345	**-1.8013**	$\mathbf{8.1 \times 10^{-7}}$	**-1.9998**

This is a linear difference equation with constant coefficient. To compute parameters (a_j, b_i) of system, we construct an energy function as follows

$$E = \left[\sum_{j=0}^{n} a_{n-j} y(k-j) - \sum_{i=0}^{m} b_{m-i} x(k-i)\right]^2 \tag{20}$$

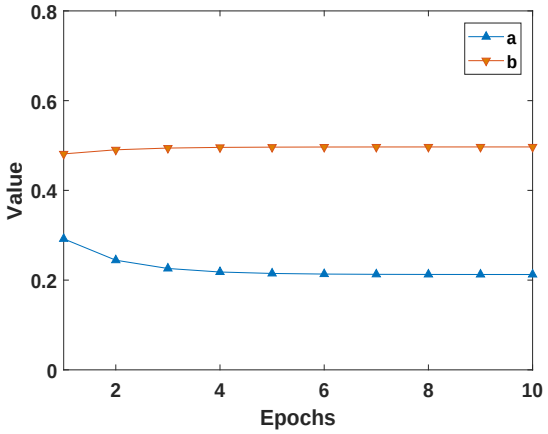


Figure 8: The estimated parameter value of the system response function

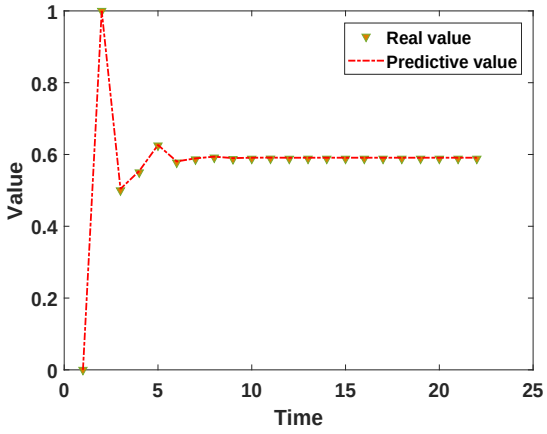


Figure 9: Output data of the response system of predictive value and real value

We substitute input and output data (x, y) and minimize the energy function by adjusting parameters. Furthermore, we utilize our proposed DE-FGD algorithm to compute iteratively.

$$a_j^{t+1} = a_j^t - \lambda(a_j^t - a_j^\star) - \mu \mathrm{D}_{a_j}^\alpha E \tag{21}$$

$$b_i^{t+1} = b_i^t - \lambda(b_i^t - b_i^\star) - \mu \mathrm{D}_{b_i}^\alpha E \tag{22}$$

where $(a_j^\star, b_i^\star) = \arg\min_{a_j, b_i}\{E\}$, and the partial fractional gradient are given as follows

$$\mathrm{D}_{a_j}^\alpha E = a_j^{-\alpha}\left[\frac{2y^2(k-j)a_j^2}{\Gamma(3-\alpha)} + \frac{2y(k-j)a_j K_1}{\Gamma(2-\alpha)} + \frac{K_1^2}{\Gamma(1-\alpha)}\right] \tag{23}$$

$$\mathrm{D}_{b_i}^\alpha E = b_i^{-\alpha}\left[\frac{2x^2(k-i)b_i^2}{\Gamma(3-\alpha)} + \frac{2x(k-i)b_i K_2}{\Gamma(2-\alpha)} + \frac{K_2^2}{\Gamma(1-\alpha)}\right] \tag{24}$$

where parameters K_1, K_2 are given as follows

$$K_1 = \sum_{p\neq j} a_p y(k-p) - \sum_{q=0}^{m} b_q x(k-q) \tag{25}$$

$$K_2 = \sum_{q\neq i} b_q x(k-q) - \sum_{p=0}^{n} a_p y(k-p) \tag{26}$$

Randomly picking an LTI system, we solve the parameters by utilizing our proposed method.

$$y(k) + 0.5y(k-1) + 0.2y(k-2) = x(k) \tag{27}$$

where $x(k) = \varepsilon(k)$ denotes a step signal. We estimate parameters in 10 epochs, and the result is shown in Figure 8. From Figure 8, we observe that after 4 epochs, parameters tend to be stable swiftly, and the calculated results are $a = 0.1961, b = 0.4959$. So they approximate to the actual parameters.

We substitute the estimated value into the response function formula, and the predictive and real value are drawn in Figure 9. And Figure 9 illustrates that predicted value almost coincides with the real value, which has a good approximation fitting effect. Therefore, the DE-FGD method is able to estimate the system response function parameters, so as to achieve the accurate prediction of output signals.

3.2. Parameter estimation of value state function in reinforcement learning

Different from supervised learning and unsupervised learning, reinforcement learning [21, 22] is a significant method that has gained recent research interest in the machine learning community, e.g. intelligent control and data analysis.

Reinforcement learning uses the framework of Markov decision processes (MDPs) to stipulate the interaction between a learning agent and its environment . Namely, a learning agent interacts with environment in discrete time (shown in Figure 10). At each step t. the agent receives a reward r_t. Then it chooses an action a_t from a series of available actions A. The agent moves to a new state s_{t+1} and the new reward r_{t+1} related with the transition (s_t, a_t, s_{t+1}) is determined.

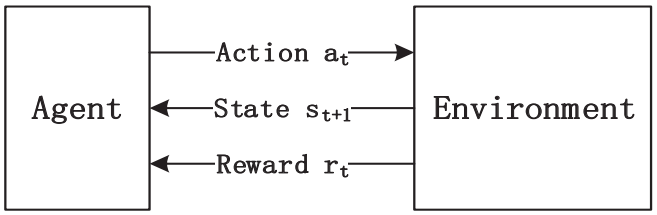


Figure 10: Interaction between a learning agent with environment

Generally speaking, reinforcement learning mainly contains model learning, model-free learning and value function approximation. In continuous space or a wide range of discrete space, the computing resource is limited. We thus could not use a table to record each value of the space location and action strategy, so we consider using parameter estimation method and construct an estimation function $V_\theta(\mathbf{x})$. By using the constructor as the approximation of the value function $V^\pi(\mathbf{x})$, the approximation is measured by the sum of squares as follows

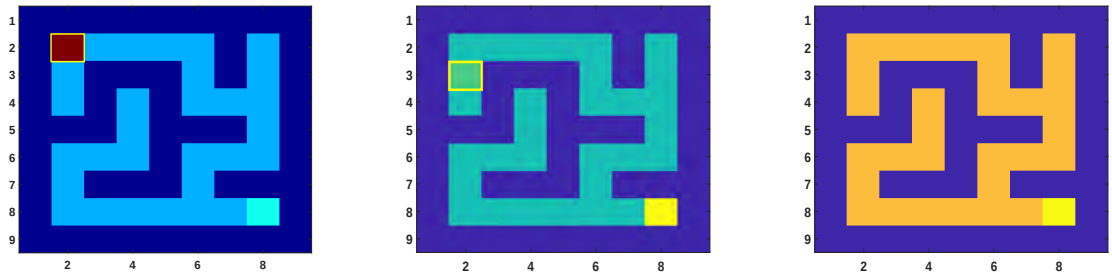
$$E_\theta = E_{\mathbf{x}\sim\pi}\left[(V^\pi(\mathbf{x}) - V_\theta(\mathbf{x}))^2\right] \tag{28}$$

where $\mathbf{x}$ is a state vector, θ is a parameter vector. To minimize Eq. 28, we utilize our method to update the estimated parameter vector as follows:

$$\theta = \theta - \lambda(\theta - \theta^\star) - \mu \mathrm{D}_\theta^\alpha E_\theta \tag{29}$$

$$\mathrm{D}_{\theta}^{\alpha} E_{\theta} = V_{\theta}(x)^{-\alpha} \left[\frac{2V_{\theta}(x)^2}{\Gamma(3-\alpha)} + \frac{2V^{\pi}(x)V_{\theta}(x)}{\Gamma(2-\alpha)} + \frac{V^{\pi}(x)^2}{1-\alpha} \right] \mathrm{D}_{\theta}^{\alpha} V_{\theta}(x) \tag{30}$$

For a specific instance of reinforcement learning – robot seeking target in a maze [23]. Firstly, the robot prepares in the starting position (2, 2) (Figure 11 a). Then, it performs an action (up, down, left and right) in the state space (Figure 11 b), with getting a series of feedback rewards. Eventually, the robot reaches destination (8, 8) (Figure 11 c).



(a) (b) (c)

Figure 11: The diagram of robot searching for the target in the maze (a: ready, b: action, c: termination)

Compared with model free learning method [24] recording 81 (9×9) state values, our method only records a parameter vector θ. It thus economizes memory resources. And satisfactory global search competence ensures that robot converges to the target more quickly.

Conclusion

To deal with the limitations of the current state of the art optimization algorithms, we propose a novel differential evolution algorithm based on adaptive fractional gradient descent (DE-FGD). Furthermore, in order to validate the performance of our proposed method, we explore an appropriate order for solving specific problems and compare our DE-FGD algorithm with state of the art optimizers (PSO, DE and GSA). It is evident that our proposed method is able to obtain the optimal solution more precisely with the order $\alpha = 1.6$ than other orders. Eventually, we apply our method into specific applications – estimating parameters of the system response function and approximate function, to predict the output value and improve the efficiency of the robot seeking for the target.

Acknowledgments

We would like to express our gratitude to the all reviewers for their very valuable and insightful remarks and comments.

References

[1] S. Mirjalili, A. Lewis, S. Sadiq, Autonomous particles groups for particle swarm optimization, Arabian Journal for Science and Engineering 39 (2014) 4683–4697.
[2] R. Fletcher, C. Reeves, Function minimization by conjugate gradients, Computer Journal 7 (1964) 149–154.
[3] T. Yang, L. Zhang, Efficient Stochastic Gradient Descent for Strongly Convex Optimization, Computer Science 50 (2013) 139–151.
[4] L. Bottou, Large-Scale Machine Learning with Stochastic Gradient Descent, Proceedings of COMPSTAT'2010, 2010.
[5] J. Maclean, J. Tsotsos, Fast Pattern Recognition Using Gradient-Descent Search in an Image Pyramid, International Conference on Pattern Recognition, 2000. Proceedings 2 (2000) 873–877.
[6] P. Zhou, Z. Liu, X. Wang, Y. Ma, H. Ma, X. Xu, S. Guo, Coherent beam combining of fiber amplifiers using stochastic parallel gradient descent algorithm and its application, IEEE journal of selected topics in quantum electronics 15 (2009) 248–256
[7] H. Yang, S. Amari, Complexity Issues in Natural Gradient Descent Method for Training Multilayer Perceptrons, Neural Computation 10 (1998) 2137–2157.

[8] S. Olariu, Y. Zomaya, Handbook Of Bioinspired Algorithms And Applications, Chapman & Hall/CRC, 2006
[9] J. Kennedy, R Eberhart, Particle swarm optimization, Proc. of 1995 IEEE Int. Conf. Neural Networks 4 (2011) 1942–1948.
[10] M. Dorigo, L. Gambardella, Ant Colony System: A cooperative learning approach to the traveling salesman problem, IEEE Transactions on Evolutionary Computation 1 (1997) 53–66.
[11] J. Koehler, Conditions that Obviate the No-Free-Lunch Theorems for Optimization, Informs Journal on Computing 19 (2007) 273–279.
[12] E. Rashed, H. Nezamabadi-Pour and S. Saryazdi, GSA: a gravitational search algorithm, Information sciences 179 (2009) 2232–2248.
[13] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, Journal of global optimization 11 (1997) 341–359.
[14] F. Brown, D. Pietra, L. Mercer, The mathematics of statistical machine translation: parameter estimation, Computational Linguistics 19 (1993) 263–311.
[15] G. Lagoudaki, Value Function Approximation, Springer US, 2017.
[16] B. Du, Y. Chen, Y. Wei, S. Cheng, Y. Wang, Discussion on extreme points with fractional order derivatives, Control Conference, 2016.
[17] Y. Pu, J. Zhou, Y. Zhang, N. Zhang, G. Huang, P. Siarry, Fractional Extreme Value Adaptive Training Method: Fractional Steepest Descent Approach, IEEE Transactions on Neural Networks & Learning Systems 26 (2015) 653–662.
[18] M. Torge, R. Bottlender, A. Strauβ, J. Möller, An introduction to the fractional calculus and fractional differential equations, Wiley, 1993.
[19] Y. Tan, Z. He, B. Tian, A Novel Generalization of Modified LMS Algorithm to Fractional Order, IEEE Signal Processing Letters 22 (2015) 1244–1248.
[20] K. Peng, Q. Lang, A. Billings, R. Tomlinson, Comparisons between harmonic balance and nonlinear output frequency response function in nonlinear system analysis, Journal of Sound & Vibration 311 (2008) 56–73.
[21] L. Busoniu, R. Babuska, D. Schutter, D. Ernst, Reinforcement Learning and Dynamic Programming Using Function Approximators, CRC Press, Inc, 2010.
[22] R. Sutton, A. Barto, Reinforcement Learning: An Introduction, Bradford Book, IEEE Transactions on Neural Networks 16 (2005) 285–286.
[23] F. Uwano, K. Takadama, Communication-Less Cooperative Q-Learning Agents in Maze Problem, Springer International Publishing, 2017.
[24] J. Perkins, D. Pendrith, On the Existence of Fixed Points for Q-Learning and Sarsa in Partially Observable Domains, Nineteenth International Conference on Machine Learning, 2002.