International Association of
Swarm and Evolutionary Intelligence

Advances in Swarm Intelligence 1 (2019) 3–11

Generative Adversarial Optimization

Ying Tan[1], Bo Shi[1]

Key Laboratory of Machine Perception (Ministry of Education),
School of Electronics Engineering and Computer Science,
Peking University, Beijing, 100871, China

Abstract

Inspired by the adversarial learning in generative adversarial network, a novel optimization framework named Generative Adversarial Optimization (GAO) is proposed in this paper. This GAO framework sets up generative models to generate candidate solutions via an adversarial process, in which two models are trained alternatively and simultaneously, i.e., a generative model for generating candidate solutions and a discriminative model for estimating the probability that a generated solution is better than a current solution. The training procedure of the generative model is to maximize the probability of the discriminative model. Specifically, the generative model and the discriminative model are in this paper implemented by multi-layer perceptrons that can be trained by the back-propagation approach. As of an implementation of the proposed GAO, for the purpose of increasing the diversity of generated solutions, a guiding vector ever introduced in guided fireworks algorithm (GFWA) has been employed here to help constructing generated solutions for the generative model. Experiments on CEC2013 benchmark suite show that the proposed GAO framework achieves better than the state-of-art performance on multi-modal functions.

Key Words: Generative Adversarial Optimization (GAO), Adversarial Learning, Generative Adversarial Network (GAN), Guiding Vector, Multi-modal Functions

1. Introduction

Continuously-valued function optimization problem

In order to solve the problem, more and more meta-heuristic algorithms have been proposed. Meta-heuristic algorithms are usually inspired by biological or human behaviors. By designing a sophisticated mechanism to guide algorithms to find solutions, so as to avoid local optimal solutions and find global optimal solutions. The most critical component for meta-heuristic algorithms is generating solutions and retaining solutions. For the part of generating solutions, the algorithm should generate better solutions as many as possible, but at the same time, it is also hoped that the generated solutions have a rich diversity and will not cluster in local optimal spaces. For the part of retaining solutions, the algorithm should retain better solutions, but it is also hoped that potential solutions which are not so good currently can be retained, because solutions which is better than the current optimal solution may be found in the local searches around them later.

In the early meta-heuristic algorithms, various methods to generate solutions were proposed. Particle swarm optimization (PSO)

In recent years, generative adversarial network (GAN)

Inspired by the adversarial learning in GAN, a feasible optimization framework, so-called Generative Adversarial Optimization (GAO), is proposed in this paper. The framework sets up generative models to generate candidate solutions via an adversarial process, in which two models are trained alternatively and simultaneously, i.e., a generative model $\mathcal{G}$ to generate candidate solutions, and a discriminative model $\mathcal{D}$ to estimates the probability that a generated solution is better than a current solution. The training procedure for $\mathcal{G}$ is to maximize the probability of $\mathcal{D}$. In our case, $\mathcal{G}$ and $\mathcal{D}$ are defined by multi-layer perceptrons, which can be trained with back-propagation. To improve the quality of generated solutions, the guiding vectors introduced in GFWA are employed to help constructing generated solutions. Experiments on CEC2013 benchmark suite show that the proposed framework achieves impressive performance on multi-modal functions.

The main contributions of this paper are as follows:

1. Inspired by adversarial learning and GAN, a novel optimization framework so-called Generative Adversarial Optimization, GAO for short, is proposed.
2. The guiding vectors introduced in GFWA
   The remainder of this paper is organized as follows. Section 2 presents related works of meta-heuristic algorithms

Email: ytan@pku.edu.cn (Ying Tan), pkushibo@pku.edu.cn (Bo Shi)

and GAN. Section 3 describes the detail of GAO, a novel optimization framework proposed for continuously-valued function optimization. Experimental settings and results are presented and discussed in Section 4. Conclusions are given in Section 5.

2. Related Works

2.1. Meta-heuristic Algorithms

Inspired by biological and human behaviors, meta-heuristic algorithms are a kind of algorithms that can be used to better solve continuous optimization problems by simulating agents' behaviors in order to balance "exploration" and "exploitation"

Swarm intelligence algorithms are usually inspired by the behavior of biological groups in natural world to seek the optimum in search space by employing programs to simulate the interaction among biological individuals. Swarm intelligence algorithms mainly focus on biological groups such as ant colony

Evolutionary computation algorithms are primarily inspired by biological evolution, which solves the global optimal solution by simulating the evolution of organisms. Specific algorithms include genetic algorithm (GA)

2.2. Generative Adversarial Networks

Generative adversarial network (GAN), which was first proposed by Ian Goodfellow in 2014

Since GAN was proposed, it has quickly become a hot research issue. A large number of researches based on GAN have sprung up, mainly focusing on optimizing GAN's structure

3. GAO: Generative Adversarial Optimization

GAO and its detailed implementation are presented in this section. First, the model architectures are described in Section 3.1, then the training procedure of GAO is discussed in details in Section 3.2.

3.1. Model Architectures

Different from the existing meta-heuristic algorithms which mainly adopt random sampling to generate elite solutions or guiding vectors

Given a objective function f, an optimization problem seeks to find the global minimum $x_* \in A$ which satisfies:

$$f(x_*) \leq f(x)\ ,\ \forall x \in A \tag{1}$$

where A is the searching space.

As illustrated in Figure 1, $\mathcal{G}$ gets the input, which includes a current solution x_c, a noise z and a step size l, and outputs a guiding vector g. This procedure can be expressed in Equation 2:

$$g = \mathcal{G}(x_c, z, l) \tag{2}$$

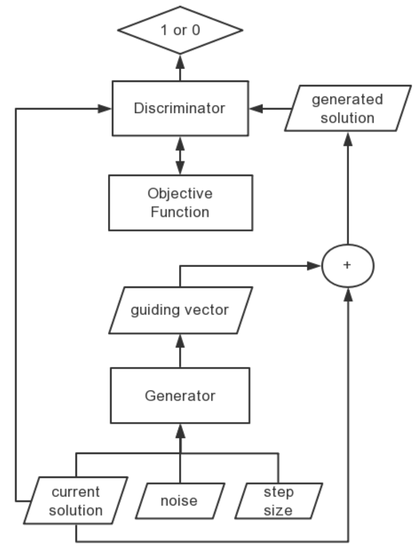

Figure 1: Architecture of GAO

Then the guiding vector g is added to the current solution x_c to get the generated solution x_g, as shown in Equation 3:

$$x_g = x_c + g \tag{3}$$

$\mathcal{D}$ receives a current solution x_c and a generated solution x_g, then outputs a prediction p that whether the generated solution x_g is better than the current solution x_c as shown in Equation 4. If the generated solution x_g is better than the current solution x_c, let $p = 1$, otherwise $p = 0$.

$$p = \mathcal{D}(x_c, x_g) = \begin{cases} 1,\ x_g\ is\ better\ than\ x_c \\ 0,\ else \end{cases} \tag{4}$$

In order to train $\mathcal{D}$, labels y^i for tuples of current solution and generated solution $\{x_c^i, x_g^i\}$ are required. The objective function f is employed to label the two-tuple set $\{x_c^i, x_g^i\}$ as expressed in Equation 5. The training of $\mathcal{D}$ will be detailedly discussed in Section 3.2.

$$y^i = \begin{cases} 1,\ if\ f(x_g^i) < f(x_c^i) \\ 0,\ else \end{cases} \tag{5}$$

The architecture of $\mathcal{G}$ is illustrated in Figure 2. First, $\mathcal{G}$ concatenate the current solution x_c and noise z included in the input, then feed the concatenated vector to a fully-connected layer (denoted as FC). Finally $\mathcal{G}$ dot the concatenated vector with step size l and get the guiding vector g as $\mathcal{G}$'s output. This procedure can be expressed in Equation 6.

$$g = \mathcal{G}(x_c, z, l) = FC([x_c^T, z^T]^T) \cdot l \tag{6}$$

The architecture of $\mathcal{D}$ is illustrated in Figure 3. First, $\mathcal{D}$ feed two solutions x_c, x_g to the same fully-connected layer denoted as FC_1, then subtract the output of x_c with the output of x_g. Finally, $\mathcal{D}$ feed the subtracted vector to

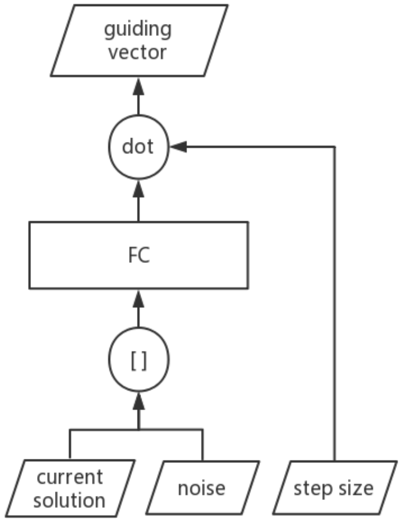

Figure 2: Architecture of $\mathcal{G}$

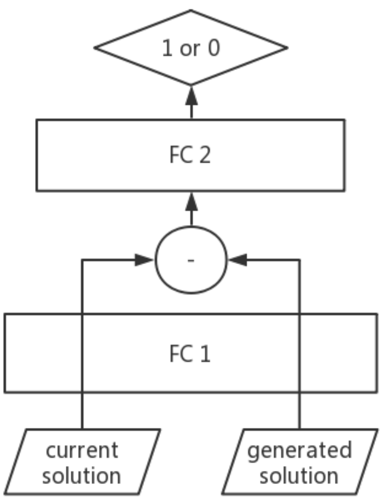

Figure 3: Architecture of $\mathcal{D}$

a fully-connected layer denoted as FC_2 and get the prediction p as $\mathcal{D}$'s output. This procedure can be expressed in Equation 7. The activation function for the final layer of FC_2 should be sigmoid function to regularize the prediction.

$$p = \mathcal{D}(x_c, x_g) = FC_2(FC_1(x_c) - FC_1(x_g)) \qquad (7)$$

3.2. Training of GAO

The complete training procedure of GAO is shown in Algorithm 1. At the beginning, μ solutions are randomly sampled in searching space to make up the solution set $C = \{x_c^i\ ,\ i = 1, 2, ..., \mu\}$, calculate each solution's fitness value $f(x_c^i)$ and initialize the step size l. Then we repeatedly do adversarial training of $\mathcal{D}$ and $\mathcal{G}$, select solutions to be retained and reduce step size l as the iteration progresses. When the termination criterion is met, the algorithm exit the loop. Since the time allowed to evaluate the solution using fitness function is limited as $MaxFES = 10000 * D$, in which D is the evaluation dimension of fitness function, the termination criterion always refers to whether the limited evaluation time is used up. Details of training $\mathcal{D}$ and $\mathcal{G}$, selecting solutions and reducing step size are discussed below.

Algorithm 1 Training procedure of GAO

Require: μ: number of current solutions
Require: β: number of solutions generated at each iteration
Require: l_{init}: initial value of step size l
1: randomly sample μ solutions in searching space A as set $C = \{x_c^i\}$
2: calculate fitness value $f(x_c^i)$ for each solution x_c^i in C
3: initialize the step size $l = l_{init}$
4: **while** termination criterion is not met **do**
5: generate β solutions and train $\mathcal{D}$
6: train $\mathcal{G}$ with fitted $\mathcal{D}$
7: select μ solutions for next iteration from μ current solutions and β generated solutions
8: reduce step size l
9: **end while**

3.2.1. Training of $\mathcal{D}$

$\mathcal{D}$ is trained to evaluate whether the generated solution x_g will be better than the current solution x_c. Train $\mathcal{D}$ requires employing $\mathcal{G}$ to generate solutions first. In this paper, the number of solutions to be generated totally at each iteration is denoted as β. Since $\mathcal{D}$ receives two solutions as input and output a prediction, training $\mathcal{D}$ requires triplets composed of two solutions x_c^i and x_g^i and a label y^i, in which y^i can be calculated with Equation 5. For a triplet $\{x_c^i, x_g^i, y^i\}$, the loss function of $\mathcal{D}$ can be calculated with Equation 8:

$$\max_{\mathcal{D}}\ loss_{\mathcal{D}} = y^i \log(D(x_c^i, x_g^i)) + (1 - y^i)\log(1 - D(x_c^i, x_g^i)) \qquad (8)$$

When training with batches, the loss of a batch is the average loss for each triplet in batch.

3.2.2. Training of $\mathcal{G}$

As mentioned above, $\mathcal{G}$ learns how to generate better guiding vectors under the guidance of $\mathcal{D}$, which means that $\mathcal{G}$ is trained by computing gradients from the feedback of $\mathcal{D}$. $\mathcal{G}$ is trained to generate elite guiding vectors for current solutions, so it's hoped that the generated solutions perform better than current solutions. For a current solution x_c^i, the loss function of $\mathcal{G}$ can be calculated with Equation 9:

$$\max_{\mathcal{G}}\ loss_{\mathcal{G}} = \log(D(x_c^i, x_c^i + \mathcal{G}(x_c^i, z, l))) \qquad (9)$$

In which, z is a random Gaussian noise, l is the step size. When training with batches, the loss of a batch is the average loss for each triplet in the batch.

3.2.3. Selecting Solutions

In general, solutions with better fitness values should be retained, so we calculate the probability to be selected for each solution x^i in Equation 10 and select solutions using the calculated probability:

$$p_r(x^i) = \frac{\gamma_{f(x^i)^{-\alpha}}}{\sum_{i=1}^{n} \gamma_{f(x^i)}^{-\alpha}} \tag{10}$$

where $\gamma_{f(x^i)}$ means the rank of fitness value for x^i among all solutions, n is the total number of candidate solutions, α is a hyper-parameter to control the shape of the distribution. The larger α is, the probability of solutions with better fitness values is larger as well.

3.2.4. Reducing Step Size

In GAO, the guiding vector introduced in GFWA

4. Experiments

In this section, principles on how to set parameters and construct $\mathcal{D}$ and $\mathcal{G}$ are given. In more detail, we first introduce the model architecture specifically and give principles for setting parameters. Secondly, the benchmark the experiment taken on is introduced. Finally, we compare GAO with other famous optimization algorithms.

In our experiment, the architecture of $\mathcal{D}$ and $\mathcal{G}$ are mainly fully-connected layers. In this section, we denote the number of hidden layers as L, the sizes of each hidden layer as H, the sizes of output layer as O ,the activation functions of each hidden layer as AH and the activation functions of output layer as AO. each of them is introduced respectively as follows. For FC in $\mathcal{G}$, we set $L = 1$, $H = [64]$, $O = dimension\ of\ objective\ function$, $AH = [relu]$, $AO = tanh$. For $FC1$ in $\mathcal{D}$, we set $L = 2$, $H = [64, 64]$, $O = 10$, $AH = [relu, relu]$, $AO = relu$. For $FC2$ in $\mathcal{D}$, we set $L = 1$, $H = [10]$, $O = 1$, $AH = [relu]$, $AO = sigmoid$.

The number of solutions retained at each iteration is deneted as μ, which mainly keeps the balance between "exploration" and "exploitation"

To train $\mathcal{D}$, we need to label the tuple of $\{x_c^i, x_g^i\}$ with y^i, which requires using objective function to evaluate the fitness value of x_g^i, since fitness value of x_c^i have been calculated at the former iteration. To make $\mathcal{D}$ learn how to generate solutions better, we not only generate x_g^i from $\mathcal{G}$, but also generate x_g^i from local search and global search at each iteration. When generating solution, x_g^i calculated from Equation 3 have to be clipped to the boundary once it exceeds the search space. In this paper, we denote the number of solutions to be generates totally at each iteration as β. On account of the limit of $MaxFES$, the iteration number $MaxIter = \frac{MaxFES}{\beta}$. In this paper, we set $\beta = 30$.

As discussed in Section 3.2.3, when selecting solutions, we calculate a probability to be selected for each solution x^i as expressed in Equation 10 and select solutions in accordance with that probability. We denote the parameter controlling the shape of the distribution as α. The larger α is, the probability of solutions with better fitness values is larger as well. In this paper, we set $\alpha = 2$ as suggested in

In our experiment, step size l have to be set as l_{init} at the beginning of the algorithm. In general, we set $l_{init} = \frac{1}{2} \cdot radius\ of\ search\ space$. Specifically for CEC2013

We compare different monotone functions on CEC2013 benchmark suite and the average ranks (ARs) are shown in Figure 5 and Figure 6, in which AR-uni, AR-multi and AR-all indicate average ranks for uni-modal, multi-modal and all functions respectively. It shows that using power function performs better than exponential function and using 4.5 as power is comprehensively best. In this paper, we use power function and set power to 4.5.

We choose CEC2013 single objective optimization benchmark suite

We compared GAO with the famous optimization algorithms including the artificial bee colony algorithm (ABC), the standard particle swarm optimization 2011 (SPSO2011)

As illustrated in Table 1, on all functions, IPOP-CMA-ES performs best, followed by GAO and LoT-FWA, while SPSO2011 is the worst one. IPOP-CMA-ES, ABC, GAO and LoT-FWA achieve 11, 10, 6 and 5 of 28 minimal mean errors on all functions, respectively. Specifically on uni-modal functions, IPOP-CMA-ES performs best as well, followed by DE, SPSO2011 and GAO performing comparable, while ABC is the worst one. IPOP-CMA-ES achieves all minimal errors on uni-modal functions, while ABC, SPSO2011, LoT-FWA and GAO achieve 1 of 5 the minimal mean errors.

On multi-modal functions, GAO performs best, followed by LoT-FWA and IPOP-CMA-ES, while SPSO2011 is the worst one. ABC achieves 8 of 23 minimal mean errors on multi-modal functions, followed by IPOP-CMA-ES, GAO and LoT-FWA, achieving 6, 5, 4 of 23 minimal mean errors, respectively. SPSO2011 and DE performs worst, achieving none minimal mean errors on multi-modal functions. Although ABC achieves 8 minimal mean errors on multi-modal functions, it also achieves 10 maximal mean error, which shows that ABC is not stable enough. At the same time, GAO achieves none maximal mean errors on all functions, which shows that GAO is quite stable and can be adapted to various problems.

It turns out from the experimental results that the proposed GAO framework performs quite very well on multi-modal functions. This is mainly due to the adversarial learning procedure, which enables $\mathcal{G}$ to learn how to generate elite and diverse solutions under the supervision of $\mathcal{D}$, rather than to follow an artificially-designed meta-heuristic rule directly. In our implementation, the guiding vector introduced in GFWA

Table 1: Mean error, standard variance and average ranks of the chosen algorithms on CEC2013 benchmark suite

	ABC		SPSO2011		IPOP-CMAES		DE		LoT-FWA		GAO	
CEC2013	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
1	**0.00E+00**	0.00E+00	**0.00E+00**	1.88E-13	**0.00E+00**	0.00E+00	1.89E-03	4.65E-04	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
2	6.20E+06	1.62E+06	3.38E+05	1.67E+05	**0.00E+00**	0.00E+00	5.52E+04	2.70E+04	1.19E+06	4.27E+05	1.02E+06	6.89E+05
3	5.74E+08	3.89E+08	2.88E+08	5.24E+08	**1.73E+00**	9.30E+00	2.16E+06	5.19E+06	2.23E+07	1.91E+07	7.98E+06	1.01E+07
4	8.75E+04	1.17E+04	3.86E+04	6.70E+03	**0.00E+00**	0.00E+00	1.32E-01	1.02E-01	2.13E+03	8.11E+02	3.17E+03	1.49E+03
5	**0.00E+00**	0.00E+00	5.42E-04	4.91E-05	**0.00E+00**	0.00E+00	2.48E-03	8.16E-04	3.55E-03	5.01E-04	2.95E-03	4.70E-04
AR. uni	4		3.4		1		3.2		3.8		3.4	
6	1.46E+01	4.39E+00	3.79E+01	2.83E+01	**0.00E+00**	0.00E+00	7.82E+00	1.65E+01	1.45E+01	6.84E+00	1.73E+01	1.53E+01
7	1.25E+02	1.15E+01	8.79E+01	2.11E+01	1.68E+01	1.96E+01	4.89E+01	2.37E+01	5.05E+01	9.69E+00	**1.08E+01**	8.15E+00
8	2.09E+01	4.97E-02	2.09E+01	5.89E-02	2.09E+01	5.90E-02	2.09E+01	5.65E-02	2.09E+01	6.14E-02	**2.09E+01**	6.96E-02
9	3.01E+01	2.02E+00	2.88E+01	4.43E+00	2.45E+01	1.61E+01	1.59E+01	2.69E+00	1.45E+01	2.07E+00	**1.09E+01**	2.10E+00
10	2.27E-01	6.75E-02	3.40E-01	1.48E-01	**0.00E+00**	0.00E+00	3.24E-02	1.97E-02	4.52E-02	2.47E-02	1.09E-02	1.05E-02
11	**0.00E+00**	0.00E+00	1.05E+02	2.74E+01	2.29E+00	1.45E+00	7.88E+01	2.51E+01	6.39E+01	1.04E+01	5.84E+01	1.84E+01
12	3.19E+02	5.23E+01	1.04E+02	3.54E+01	**1.85E+00**	1.16E+00	8.14E+01	3.00E+01	6.82E+01	1.45E+01	5.60E+01	1.26E+01
13	3.29E+02	3.91E+01	1.94E+02	3.86E+01	**2.41E+00**	2.27E+00	1.61E+02	3.50E+01	1.36E+02	2.30E+01	1.09E+02	2.61E+01
14	**3.58E-01**	3.91E-01	3.99E+03	6.19E+02	2.87E+02	2.72E+02	2.38E+03	1.42E+03	2.38E+03	3.13E+02	2.38E+03	4.39E+02
15	3.88E+03	3.41E+02	3.81E+03	6.94E+02	**3.38E+02**	2.42E+02	5.19E+03	5.16E+02	2.58E+03	3.83E+02	2.43E+03	4.78E+02
16	1.07E+00	1.96E-01	1.31E+00	3.59E-01	2.53E+00	2.73E-01	1.97E+00	2.59E-01	**5.74E-02**	2.13E-02	7.72E-02	4.24E-02
17	**3.04E+01**	5.15E-03	1.16E+02	2.02E+01	3.41E+01	1.36E+00	9.29E+01	1.57E+01	6.20E+01	9.45E+00	9.40E+01	1.80E+01
18	3.04E+02	3.52E+01	1.21E+02	2.46E+01	8.17E+01	6.13E+01	2.34E+02	2.56E+01	**6.12E+01**	9.56E+00	8.92E+01	2.64E+01
19	**2.62E-01**	5.99E-02	9.51E+00	4.42E+00	2.48E+00	4.02E-01	4.51E+00	1.30E+00	3.05E+00	6.43E-01	3.68E+00	8.05E-01
20	1.44E+01	4.60E-01	1.35E+01	1.11E+00	1.46E+01	3.49E-01	1.43E+01	1.19E+00	1.33E+01	1.02E+00	**1.10E+01**	6.91E-01
21	**1.65E+02**	3.97E+01	3.09E+02	6.80E+01	2.55E+02	5.03E+01	3.20E+02	8.55E+01	2.00E+02	2.80E-03	2.94E+02	6.29E+01
22	**2.41E+01**	2.81E+01	4.30E+03	7.67E+02	5.02E+02	3.09E+02	1.72E+03	7.06E+02	3.12E+03	3.79E+02	2.99E+03	5.34E+02
23	4.95E+03	5.13E+02	4.83E+03	8.23E+02	**5.76E+02**	3.50E+02	5.28E+03	6.14E+02	3.11E+03	5.16E+02	2.67E+03	6.25E+02
24	2.90E+02	4.42E+00	2.67E+02	1.25E+01	2.86E+02	3.02E+01	2.47E+02	1.54E+01	**2.37E+02**	1.20E+01	2.39E+02	6.32E+00
25	3.06E+02	6.49E+00	2.99E+02	1.05E+01	2.87E+02	2.85E+01	2.80E+02	1.57E+01	2.71E+02	1.97E+01	**2.60E+02**	1.74E+01
26	2.01E+02	1.93E-01	2.86E+02	8.24E+01	3.15E+02	8.14E+01	2.52E+02	6.83E+01	**2.00E+02**	1.76E-02	2.00E+02	4.32E-02
27	**4.16E+02**	1.07E+02	1.00E+03	1.12E+02	1.14E+03	2.90E+02	7.64E+02	1.00E+02	6.84E+02	9.77E+01	6.45E+02	5.58E+01
28	**2.58E+02**	7.78E+01	4.01E+02	4.76E+02	3.00E+02	0.00E+00	4.02E+02	3.90E+02	2.65E+02	7.58E+01	2.96E+02	2.77E+01
AR. multi	3.57		4.87		2.87		4.04		2.61		2.48	
AR. all	3.64		4.61		2.54		3.89		2.82		2.64	

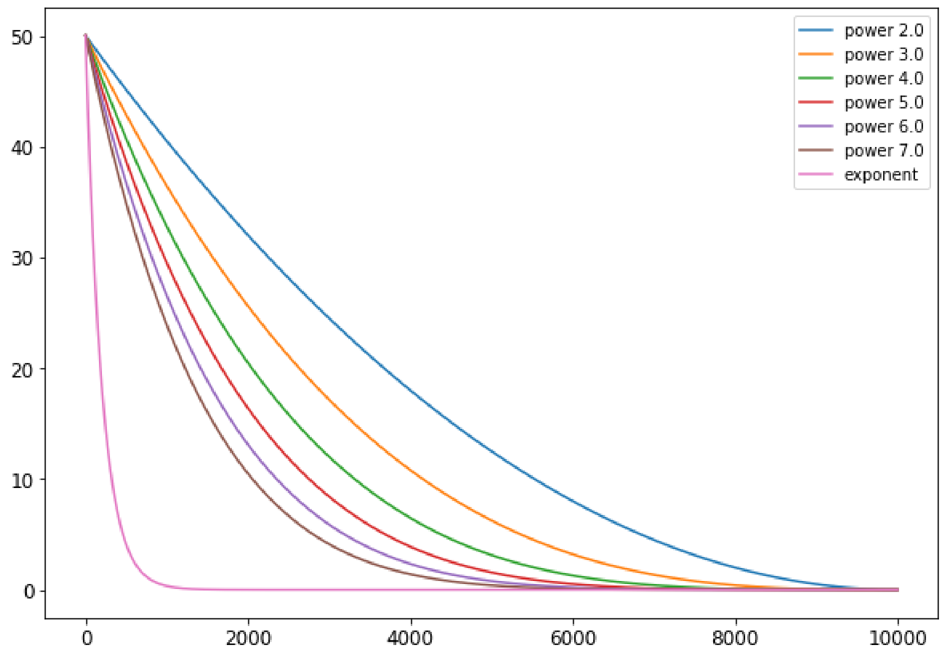

Figure 4: how step size changes with different monotone functions

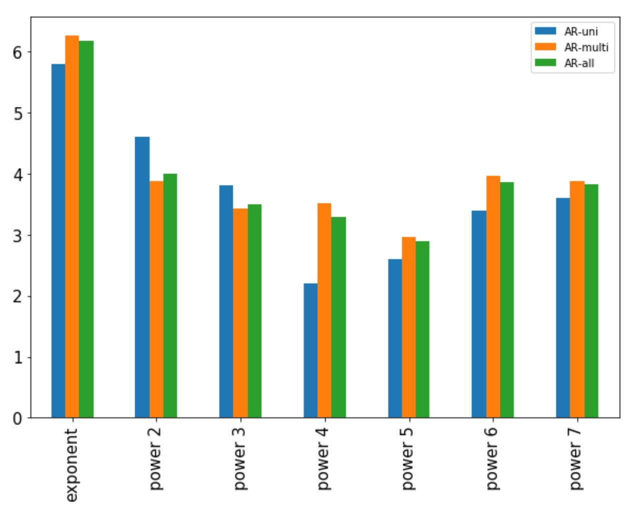

Figure 5: Average ranks for different monotone functions

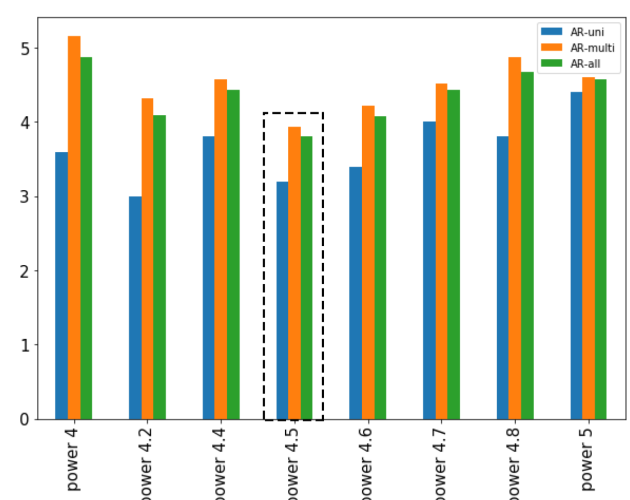

Figure 6: Average ranks for power function with different power

5. Conclusion

Inspired by the adversarial learning in generative adversarial network, this paper proposed a novel optimization framework, so-called GAO for short, which is the first attempt to employ adversarial learning for continuously-valued function optimization. In order to improve the quality of generated solutions, a guiding vector appeared in GFWA is employed in this paper to help constructing generated solutions. Experiments on CEC2013 benchmark suite shew that the proposed GAO algorithm performs quite well, especially on multi-modal functions, it gave the best performance over some famous optimization approaches. Meanwhile, the performance of the GAO framework on uni-modal functions indicates that there is still room for improvement. It is worth noting that the proposed GAO framework should be further studied since it can be easily embedded into any iterative algorithms as an operator to generate solutions. We hope the this paper can be regarded as a start point to attract more research on solving various optimization problems using adversarial learning strategy.

Acknowledgement.

This work was supported by the Natural Science Foundation of China (NSFC) under grant no. 61673025 and 61375119 and also Supported by Beijing Natural Science Foundation (4162029), and partially supported by National Key Basic Research Development Plan (973 Plan) Project of China under grant no. 2015CB352302.

References

[1] Arjovsky, M., Bottou, L.: Towards principled methods for training generative adversarial networks. arxiv (2017)

[2] Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein gan. arXiv preprint arXiv:1701.07875 (2017)

[3] Auger, A., Hansen, N.: A restart cma evolution strategy with increasing population size. In: 2005 IEEE congress on evolutionary computation. vol. 2, pp. 1769–1776. IEEE (2005)

[4] Bertsekas, D.P.: Constrained optimization and Lagrange multiplier methods. Academic press (2014)

[5] Bousmalis, K., Silberman, N., Dohan, D., Erhan, D., Krishnan, D.: Unsupervised pixel-level domain adaptation with generative adversarial networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3722–3731 (2017)

[6] Che, T., Li, Y., Jacob, A.P., Bengio, Y., Li, W.: Mode regularized generative adversarial networks. arXiv preprint arXiv:1612.02136 (2016)

[7] Che, T., Li, Y., Zhang, R., Hjelm, R.D., Li, W., Song, Y., Bengio, Y.: Maximum-likelihood augmented discrete generative adversarial networks. arXiv preprint arXiv:1702.07983 (2017)

[8] Chen, J., Xin, B., Peng, Z., Dou, L., Zhang, J.: Optimal contraction theorem for exploration–exploitation tradeoff in search and optimization. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans **39**(3), 680–691 (2009)

[9] Chongxuan, L., Xu, T., Zhu, J., Zhang, B.: Triple generative adversarial nets. In: Advances in neural information processing systems. pp. 4088–4098 (2017)

[10] Chongxuan, L., Xu, T., Zhu, J., Zhang, B.: Triple generative adversarial nets. In: Advances in neural information processing systems. pp. 4088–4098 (2017)

[11] Clerc, M.: Standard particle swarm optimisation from 2006 to 2011. Particle Swarm Central **253** (2011)

[12] Conti, E., Madhavan, V., Such, F.P., Lehman, J., Stanley, K., Clune, J.: Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. In: Advances in Neural Information Processing Systems. pp. 5032–5043 (2018)

[13] Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., Bharath, A.A.: Generative adversarial networks: An overview. IEEE Signal Processing Magazine **35**(1), 53–65 (2018)

[14] Dai, W., Doyle, J., Liang, X., Zhang, H., Dong, N., Li, Y., Xing, E.P.: Scan: Structure correcting adversarial network for chest x-rays organ segmentation. arXiv preprint arXiv:1703.08770 (2017)

[15] Davis, L.: Handbook of genetic algorithms (1991)

[16] Deb, K.: Optimization for engineering design: Algorithms and examples. PHI Learning Pvt. Ltd. (2012)

[17] Deb, K.: Multi-objective optimization. In: Search methodologies, pp. 403–449. Springer (2014)

[18] Dennis Jr, J.E., Schnabel, R.B.: Numerical methods for unconstrained optimization and nonlinear equations, vol. 16. Siam (1996)

[19] Denton, E., Gross, S., Fergus, R.: Semi-supervised learning with context-conditional generative adversarial networks. arXiv preprint arXiv:1611.06430 (2016)

[20] Denton, E.L., Chintala, S., Fergus, R., et al.: Deep generative image models using a laplacian pyramid of adversarial networks. In: Advances in neural information processing systems. pp. 1486–1494 (2015)

[21] Dorigo, M., Birattari, M.: Ant colony optimization. Springer (2010)

[22] Dorigo, M., Di Caro, G.: Ant colony optimization: a new meta-heuristic. In: Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406). vol. 2, pp. 1470–1477. IEEE (1999)

[23] Fedus, W., Goodfellow, I., Dai, A.M.: Maskgan: Better text generation via filling in the_. arXiv preprint arXiv:1801.07736 (2018)

[24] Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., Lempitsky, V.: Domain-adversarial training of neural networks. The Journal of Machine Learning Research **17**(1), 2096–2030 (2016)

[25] Ghosh, A., Kulharia, V., Namboodiri, V.P., Torr, P.H., Dokania, P.K.: Multi-agent diverse generative adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8513–8521 (2018)

[26] Goodfellow, I., Bengio, Y., Courville, A.: Deep learning. MIT press (2016)

[27] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems. pp. 2672–2680 (2014)

[28] Guimaraes, G.L., Sanchez-Lengeling, B., Outeiral, C., Farias, P.L.C., Aspuru-Guzik, A.: Objective-reinforced generative adversarial networks (organ) for sequence generation models. arXiv preprint arXiv:1705.10843 (2017)

[29] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.C.: Improved training of wasserstein gans. In: Advances in Neural Information Processing Systems. pp. 5767–5777 (2017)

[30] Guo, J., Lu, S., Cai, H., Zhang, W., Yu, Y., Wang, J.: Long text generation via adversarial training with leaked information. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)

[31] Gurumurthy, S., Kiran Sarvadevabhatla, R., Venkatesh Babu, R.: Deligan: Generative adversarial networks for diverse and limited data. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 166–174 (2017)

[32] Hand, D.J.: Data mining. Encyclopedia of Environmetrics **2** (2006)

[33] Hoang, Q., Nguyen, T.D., Le, T., Phung, D.: Mgan: Training

generative adversarial nets with multiple generators (2018)
[34] Hong, Y., Hwang, U., Yoo, J., Yoon, S.: How generative adversarial networks and their variants work: An overview. ACM Computing Surveys (CSUR) **52**(1), 10 (2019)
[35] Hsu, C.C., Hwang, H.T., Wu, Y.C., Tsao, Y., Wang, H.M.: Voice conversion from unaligned corpora using variational autoencoding wasserstein generative adversarial networks. arXiv preprint arXiv:1704.00849 (2017)
[36] Hu, W., Ying, T.: Generating adversarial malware examples for black-box attacks based on gan (2017)
[37] Intriligator, M.D.: Mathematical optimization and economic theory. SIAM (2002)
[38] Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1125–1134 (2017)
[39] Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1125–1134 (2017)
[40] Kamien, M.I., Schwartz, N.L.: Dynamic optimization: the calculus of variations and optimal control in economics and management. Courier Corporation (2012)
[41] Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of gans for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196 (2017)
[42] Kennedy, J.: Swarm intelligence. In: Handbook of nature-inspired and innovative computing, pp. 187–219. Springer (2006)
[43] Kennedy, J.: Particle swarm optimization. Encyclopedia of machine learning pp. 760–766 (2010)
[44] Kim, T., Cha, M., Kim, H., Lee, J.K., Kim, J.: Learning to discover cross-domain relations with generative adversarial networks. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70. pp. 1857–1865. JMLR. org (2017)
[45] Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. science **220**(4598), 671–680 (1983)
[46] Koziel, S., Yang, X.S.: Computational optimization, methods and algorithms, vol. 356. Springer (2011)
[47] Kusner, M.J., Hernández-Lobato, J.M.: Gans for sequences of discrete elements with the gumbel-softmax distribution. arXiv preprint arXiv:1611.04051 (2016)
[48] Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al.: Photo-realistic single image super-resolution using a generative adversarial network. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4681–4690 (2017)
[49] Lee, S.g., Hwang, U.: Seonwoo min, and sungroh yoon. a seqgan for polyphonic music generation. arXiv preprint arXiv:1710.11418 (2017)
[50] Lehman, J., Chen, J., Clune, J., Stanley, K.O.: Es is more than just a traditional finite-difference approximator. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 450–457. ACM (2018)
[51] Lehman, J., Chen, J., Clune, J., Stanley, K.O.: Safe mutations for deep and recurrent neural networks through output gradients. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 117–124. ACM (2018)
[52] Leonard, D., Van Long, N., Ngo, V.L., et al.: Optimal control theory and static optimization in economics. Cambridge University Press (1992)
[53] Li, J., Tan, Y.: Loser-out tournament-based fireworks algorithm for multimodal function optimization. IEEE Transactions on Evolutionary Computation **22**(5), 679–691 (2018)
[54] Li, J., Zheng, S., Tan, Y.: Adaptive fireworks algorithm. In: 2014 IEEE congress on evolutionary computation (CEC). pp. 3214–3221. IEEE (2014)
[55] Li, J., Zheng, S., Tan, Y.: The effect of information utilization: Introducing a novel guiding spark in the fireworks algorithm. IEEE Transactions on Evolutionary Computation **21**(1), 153–166 (2017)
[56] Liang, J., Qu, B., Suganthan, P., Hernández-Díaz, A.G.: Problem definitions and evaluation criteria for the cec 2013 special session on real-parameter optimization. Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Technical Report **201212**(34), 281–295 (2013)
[57] Lin, K., Li, D., He, X., Zhang, Z., Sun, M.T.: Adversarial ranking for language generation. In: Advances in Neural Information Processing Systems. pp. 3155–3165 (2017)
[58] Lin, K., Li, D., He, X., Zhang, Z., Sun, M.T.: Adversarial ranking for language generation. In: Advances in Neural Information Processing Systems. pp. 3155–3165 (2017)
[59] Mao, X., Li, Q., Xie, H., Lau, R.Y., Wang, Z., Paul Smolley, S.: Least squares generative adversarial networks. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2794–2802 (2017)
[60] Mardani, M., Gong, E., Cheng, J.Y., Vasanawala, S., Zaharchuk, G., Alley, M., Thakur, N., Han, S., Dally, W., Pauly, J.M., et al.: Deep generative adversarial networks for compressed sensing automates mri. arXiv preprint arXiv:1706.00051 (2017)
[61] Marler, R.T., Arora, J.S.: Survey of multi-objective optimization methods for engineering. Structural and multidisciplinary optimization **26**(6), 369–395 (2004)
[62] Van der Merwe, D., Engelbrecht, A.P.: Data clustering using particle swarm optimization. In: The 2003 Congress on Evolutionary Computation, 2003. CEC'03. vol. 1, pp. 215–220. IEEE (2003)
[63] Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y.: Spectral normalization for generative adversarial networks. arXiv preprint arXiv:1802.05957 (2018)
[64] Neshat, M., Sepidnam, G., Sargolzaei, M., Toosi, A.N.: Artificial fish swarm algorithm: a survey of the state-of-the-art, hybridization, combinatorial and indicative applications. Artificial intelligence review **42**(4), 965–997 (2014)
[65] Nguyen, A., Dosovitskiy, A., Yosinski, J., Brox, T., Clune, J.: Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In: Advances in Neural Information Processing Systems. pp. 3387–3395 (2016)
[66] Nowozin, S., Cseke, B., Tomioka, R.: f-gan: Training generative neural samplers using variational divergence minimization. In: Advances in neural information processing systems. pp. 271–279 (2016)
[67] Price, K., Storn, R.M., Lampinen, J.A.: Differential evolution: a practical approach to global optimization. Springer Science & Business Media (2006)
[68] Qin, A.K., Huang, V.L., Suganthan, P.N.: Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE transactions on Evolutionary Computation **13**(2), 398–417 (2009)
[69] Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434 (2015)
[70] Rao, S.S.: Engineering optimization: theory and practice. John Wiley & Sons (2009)
[71] Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., Lee, H.: Generative adversarial text to image synthesis. arXiv preprint arXiv:1605.05396 (2016)
[72] Reed, S.E., Akata, Z., Mohan, S., Tenka, S., Schiele, B., Lee, H.: Learning what and where to draw. In: Advances in Neural Information Processing Systems. pp. 217–225 (2016)
[73] Ruder, S.: An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747 (2016)
[74] Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training gans. In: Advances in neural information processing systems. pp. 2234–2242 (2016)
[75] Shi, H., Dong, J., Wang, W., Qian, Y., Zhang, X.: Ssgan: secure steganography based on generative adversarial networks. In: Pacific Rim Conference on Multimedia. pp. 534–544. Springer (2017)

[76] Springenberg, J.T.: Unsupervised and semi-supervised learning with categorical generative adversarial networks. arXiv preprint arXiv:1511.06390 (2015)
[77] Springenberg, J.T.: Unsupervised and semi-supervised learning with categorical generative adversarial networks. arXiv preprint arXiv:1511.06390 (2015)
[78] Storn, R., Price, K.: Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization **11**(4), 341–359 (1997)
[79] Such, F.P., Madhavan, V., Conti, E., Lehman, J., Stanley, K.O., Clune, J.: Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. arXiv preprint arXiv:1712.06567 (2017)
[80] Tan, K.C., Chiam, S.C., Mamun, A., Goh, C.K.: Balancing exploration and exploitation with adaptive variation for evolutionary multi-objective optimization. European Journal of Operational Research **197**(2), 701–713 (2009)
[81] Tan, Y.: Fireworks Algorithm. Springer (2015)
[82] Tan, Y., Zhu, Y.: Fireworks algorithm for optimization. In: International conference in swarm intelligence. pp. 355–364. Springer (2010)
[83] Trelea, I.C.: The particle swarm optimization algorithm: convergence analysis and parameter selection. Information processing letters **85**(6), 317–325 (2003)
[84] Tulyakov, S., Liu, M.Y., Yang, X., Kautz, J.: Mocogan: Decomposing motion and content for video generation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1526–1535 (2018)
[85] Tulyakov, S., Liu, M.Y., Yang, X., Kautz, J.: Mocogan: Decomposing motion and content for video generation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1526–1535 (2018)
[86] Vercellis, C.: Business intelligence: data mining and optimization for decision making. John Wiley & Sons (2011)
[87] Volkhonskiy, D., Nazarov, I., Borisenko, B., Burnaev, E.: Steganographic generative adversarial networks. arXiv preprint arXiv:1703.05502 (2017)
[88] Vondrick, C., Pirsiavash, H., Torralba, A.: Generating videos with scene dynamics. In: Advances In Neural Information Processing Systems. pp. 613–621 (2016)
[89] Vondrick, C., Pirsiavash, H., Torralba, A.: Generating videos with scene dynamics. In: Advances In Neural Information Processing Systems. pp. 613–621 (2016)
[90] Walker, J., Marino, K., Gupta, A., Hebert, M.: The pose knows: Video forecasting by generating pose futures. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 3332–3341 (2017)
[91] Walker, J., Marino, K., Gupta, A., Hebert, M.: The pose knows: Video forecasting by generating pose futures. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 3332–3341 (2017)
[92] Wang, K.P., Huang, L., Zhou, C.G., Pang, W.: Particle swarm optimization for traveling salesman problem. In: Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No. 03EX693). vol. 3, pp. 1583–1585. IEEE (2003)
[93] Wu, J., Zhang, C., Xue, T., Freeman, B., Tenenbaum, J.: Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In: Advances in neural information processing systems. pp. 82–90 (2016)
[94] Yang, D., Xiong, T., Xu, D., Huang, Q., Liu, D., Zhou, S.K., Xu, Z., Park, J., Chen, M., Tran, T.D., et al.: Automatic vertebra labeling in large-scale 3d ct using deep image-to-image network with message passing and sparsity regularization. In: International Conference on Information Processing in Medical Imaging. pp. 633–644. Springer (2017)
[95] Yang, J., Shi, X., Marchese, M., Liang, Y.: An ant colony optimization method for generalized tsp problem. Progress in Natural Science **18**(11), 1417–1422 (2008)
[96] Yu, L., Zhang, W., Wang, J., Yu, Y.: Seqgan: Sequence generative adversarial nets with policy gradient. In: Thirty-First AAAI Conference on Artificial Intelligence (2017)
[97] Zhang, X., Clune, J., Stanley, K.O.: On the relationship between the openai evolution strategy and stochastic gradient descent. arXiv preprint arXiv:1712.06564 (2017)
[98] Zhang, Y., Gan, Z., Fan, K., Chen, Z., Henao, R., Shen, D., Carin, L.: Adversarial feature matching for text generation. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70. pp. 4006–4015. JMLR. org (2017)
[99] Zhao, J., Mathieu, M., LeCun, Y.: Energy-based generative adversarial network. arXiv preprint arXiv:1609.03126 (2016)
[100] Zheng, S., Janecek, A., Li, J., Tan, Y.: Dynamic search in fireworks algorithm. In: 2014 IEEE Congress on evolutionary computation (CEC). pp. 3222–3229. IEEE (2014)
[101] Zheng, S., Janecek, A., Tan, Y.: Enhanced fireworks algorithm. In: 2013 IEEE congress on evolutionary computation. pp. 2069–2077. IEEE (2013)
[102] Zheng, S., Li, J., Janecek, A., Tan, Y.: A cooperative framework for fireworks algorithm. IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB) **14**(1), 27–41 (2017)
[103] Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: Computer Vision (ICCV), 2017 IEEE International Conference on (2017)