AdamPSO: A Particle Swarm Optimization Algorithm with Adaptive Moment Estimation

Shengran Hu

***Abstract*—Optimization problems are widespread problems in acadamia and industry, and particle swarm algorithm (PSO) is one of the algorithms for solving optimization problems based on swarm intelligence. In the application of particle swarm algorithms, the learning rate is an important parameter that affects the performance of PSO significantly. In this paper, we propose the AdamPSO algorithm, which achieves adaptive adjustment of the learning rate for each dimension during the search process by introducing the popular adaptive momentum estimation method in the gradient descent method. To my best knowledge, this paper introduces the adaptive momentum estimation method in gradient-free PSO algorithm for the first time, while this method can be conceptually applied to most PSO algorithms. Comparative experiments with conventional PSO on twelve test functions show that AdamPSO can significantly enhance the local search capability of the algorithm without affecting its global exploration capability. By design, AdamPSO introduces only two parameters to the traditional PSO, and the parameter settings are explored empirically. Finally, the paper conducts an experiment-based analysis for the adaptive behavior and explores the characteristics of the adaptive momentum estimation mechanism for the regulation of the search process under different functional scenarios.**

***Index Terms*—Particle swarm optimization (PSO), adaptive moment estimation, evolutionary computation, global optimization.**

I. Introduction

Among swarm intelligence algorithms, the particle swarm algorithm (PSO) [1], [2] is one of the important classes of optimization algorithms. Such algorithms simulate the motion process of populations such as birds to search for the global optimal solution of a function. Because of its simple framework and promising performance, PSO is often used in the optimization of complex functions [3] and real-world problems [4].

Like other optimization algorithms based on velocity update, the learning rate is an important parameter in the PSO algorithm. However, the traditional learning rate mechanism not only requires a lot of experiments to set the optimal value, but also cannot be adaptively adjusted in terms of dimensionality. In this paper, we introduce the Adaptive Moment Estimation algorithm (Adam) [5] into the framework. Jiang and Han propose a PSO and Adam hybrid algorithm based on the gradient information. Unlike it, the algorithm proposed in this paper is a black-box optimization algorithm that does not rely on gradient.

Shengran Hu was with the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, Guangdong, 518055 China e-mail: 11711011@mail.sustech.edu.cn.

Manuscript submitted March 14, 2021.

The main contributions of this paper are summarized as follows.

- In this paper, an adaptive moment estimation method is introduced to the PSO framework so that the learning rate of the PSO algorithm can be dynamically adjusted based on the update history of the dimensions. In the experimental part, the two new parameters introduced are emprically validated for their settings, and an experiment-based analysis of the adaptive behavior is also performed.
- We further proposed the AdamPSO algorithm by integrating the introduced method adaptive moment estimation method into the conventional PSO for comparison tests. It is demonstrated in experiments with twelve test functions that AdamPSO can significantly enhance the local search capability of the algorithm without affecting the global exploration capability of the algorithm

The remainder of the paper is organized as follows: in Section II, the background and details of AdamPSO are presented; in Section III, comparative experiments with twelve test functions, parameter validity experiments, and adaptive behavior analysis experiments are presented; in Section IV, the current unresolved issues and possible future work are described; and finally, a summary of the paper is given in Section V.

II. The Proposed Method

A. Background

Generally, bound constrained single objective numerical optimization can be formulated as following,

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & L \leq x \leq U \end{aligned}$$

where x is the decision variable, f is the objective function, L, U are the upper and lower bounds of the decision variable x.

The PSO algorithm uses a population of multiple particles to solve such optimization problems, where each individual i denotes a possible solution, and its position vector $\mathbf{X}_i = [x_i^1, x_i^2, ..., x_i^D]$. D is the dimensionality of the search space. Each individual also has a velocity vector $\mathbf{V}_i = [v_i^1, v_i^2, ..., v_i^D]$. In each iteration, the individual moves according to its own movement velocity. The core difference between the PSO algorithm and its variants lies in the method of updating the position and movement velocity of the individual in each generation. In the conventional PSO algorithm [2],

Algorithm 1: AdamPSO

Input : Fitness function f, Dimension D, Population size m, Boundary $ub_{D\times1}$, $lb_{D\times1}$, Max generation max_gen.

Output: Best found position $gBest$ to minimize f.

// Initialization.

$V_{max} = 0.5 \times (ub - lb)$;

for $i \leftarrow 1$ **to** m **do**

  $x_i^1 = lb + (ub - lb) \cdot rand_{D\times1}$;

  Fitness$_i = f(x_i^1)$;

  $pBest = x_i^1$;

  $v_i^1 = -V_{max} + 2 \times V_{max} \cdot rand_{D\times1}$;

  $m_i^1 = 1_{D\times1}$;

end

$gBest_{D\times1} = v_i$ where $i = argmin_i$(Fitness$_i$);

// Main loop.

for $t \leftarrow 2$ **to** *max_gen* **do**

  $w^t = w_0 - \frac{t\times(w_0-w_1)}{\text{max\_gen}}$; // Intertia weight decreases within searching.

  for $i \leftarrow 1$ **to** m **do**

    $\Delta v_i^t = c_1 rand_1 \cdot (pBest_i - x_i^{t-1}) + c_2 rand_2 \cdot (gBest - x_i^{t-1})$;

    $v_i^t = w^t v_i^{t-1} + \Delta v_i^t$;

    if $v_i^t > V_{max}$ **then**

      $v_i^t = V_{max}$;

    else if $v_i^t < -V_{max}$ **then**

      $v_i^t = -V_{max}$;

    end

    $m_i^t = \beta m_i^{t-1} + (1-\beta)(\frac{\Delta v_i^t - mean(\Delta v_i^t)}{std(\Delta v_i^t)} + 1)^2$;

    $x_i^t = x_i^{t-1} + \frac{\eta}{\sqrt{m_i^t} + \epsilon} \cdot v_i^t$;

    if $x_i^d > u_d$ **then**

      $x_i^d = u_d - (\frac{rand}{4} \times (u_d - l_d))$;

    else if $x_i^d < l_d$ **then**

      $x_i^d = l_d + (\frac{rand}{4} \times (u_d - l_d))$;

    end

    Fitness$_i = f(x_i^t)$;

    if *new personal best* **then**

      $pBest = x_i^t$;

      if *new global best* **then**

        $gBest = x_i^t$;

      end

    end

  end

end

return $gBest$.

for each individual i in the tth iteration, the process can be represented as,

$$v_i^t = w_t v_i^{t-1} + c_1 rand_1 \cdot (pBest_i - x_i^{t-1}) + c_2 rand_2 \cdot (gBest - x_i^{t-1}) \quad (1)$$

$$x_i^{t+1} = x_i^t + v_i^t \quad (2)$$

where w_t denotes the intertia weight, which decreases along with the searching [2]; c_1 and c_2 are two positive real parameters; $rand_1$ and $rand_2$ are two D-dimensional random number vectors between $[0, 1]$, updated at each iteration; $pBest$ and $gBest$ are the best positions in the history of this individual and all individuals.

The PSO algorithm update velocity based on both $pBest$ and $gBest$. In fact, such directions are similar to approximations for the gradient of a function. Using this perspective on PSO, one finds that it is actually very similar to Stochastic Gradient Descent with Momentum (SGD/M) [6], which can be expressed as,

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta) \quad (3)$$

$$\theta = \theta - v_t \quad (4)$$

where θ is the variable to be optimized; v_t is the velocity in the tth iteration; γ, η are two real parameters; $\nabla_\theta J(\theta)$ is the gradient of the current location of the optimization variable θ. Since the function value increases along with direction of the gradient, the velocity in equation (4) is $-v_t$.

In both PSO and SGD/M, the learning rate is the key to the performance. In PSO, the values of w, c_1, c_2 control the learning rate; while in SGD/M, the two parameters of γ, η control the learning rate. Not only the learning rate faces difficulties in taking values, but also the learning rate mechanism has unsolvable defects. In optimization problems, the optimization process of multiple dimensions is often not synchronized, and ideally, each dimension actually needs its own, dynamically changing learning rate [5]. Based on this idea, the Adaptive Moment Estimation (Adam) [5] algorithm is proposed, which is capable of adaptively adjusting the learning rate in each dimension based on SGD/M. The algorithm can be expressed as follows,

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)(\nabla_\theta J(\theta))^2 \quad (5)$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)\nabla_\theta J(\theta) \quad (6)$$

$$\theta = \theta - \frac{\eta}{\sqrt{m_t} + \epsilon} \cdot v_t \quad (7)$$

where θ is the variable to be optimized; v_t is the velocity in the tth iteration, also known as the first-order momentum estimation; m_t is the second-order momentum estimation in the t-th iteration, which is used to dynamically adjust the learning rate; in Eq. (5), the second-order momentum m is updated by the current gradient $\nabla_\theta J(\theta)$ is updated by the quadratic of the current gradient $\nabla_\theta J(\theta)$, and β_1 is the control parameter; Eq. (6), the velocity v_t is updated by the current gradient $\nabla_\theta J(\theta)$, and β_2 is the control parameter. Eq. (7), the current velocity v_t is adaptively adjusted by $\frac{1}{\sqrt{m_t}}$; η is the learning rate control term; ϵ is a small constant to prevent division by zero, and usually takes the value 10^{-8} [5].

B. AdamPSO

1) Adaptive Moment Estimation for PSO: Similar to the improvements of Adam for SGD/M, this paper proposes an Adaptive Moment Estimation method for PSO, which introduces adaptive learning rate adjustment into the standard

TABLE I
TWELVE TEST FUNCTIONS [7] USED IN THIS REPORT.

	Test function	D	Search Space	Global f_{min}	Name of function
Unimodal	$f_1(x) = \sum_{i=1}^{D} x_i^2$	30	$[-100, 100]^D$	0	Sphere Model
	$f_2(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	30	$[-10, 10]^D$	0	Schwefel's Problem 2.22
	$f_3(x) = \sum_{i=1}^{D} (\sum_{j=1}^{i} x_j)^2$	30	$[-100, 100]^D$	0	Schwefel's Problem 1.2
	$f_4(x) = \max_i \{\lvert x_i \rvert, 1 \le i \le D\}$	30	$[-100, 100]^D$	0	Schwefel's Problem 2.21
	$f_5(x) = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	$[-30, 30]^D$	0	Rosenbrock's Function
	$f_6(x) = \sum_{i=1}^{D} (\lfloor x_i + 0.5 \rfloor)^2$	30	$[-100, 100]^D$	0	Step Function
	$f_7(x) = \sum_{i=1}^{D} i x_i^4 + random[0, 1)$	30	$[-10, 10]^D$	0	Noisy Quartic
Multimodal	$f_8(x) = \sum_{i=1}^{D} -x_i \sin(\sqrt{x_i})$	30	$[-500, 500]^D$	-12569.5	Schwefel's Problem 2.26
	$f_9(x) = \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i) + 10]$	30	$[-5.12, 5.12]^D$	0	Rastrigin's Function
	$f_{10}(x) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_i^D x_i^2}) - \exp(\frac{1}{n}\sum_i^D \cos 2\pi x_i) + 20 + e$	30	$[-32, 32]^D$	0	Ackley
	$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}}) + 1$	30	$[-600, 600]^D$	0	Griewank
	$f_{13}(x) = \sum_{i=1}^{5} [(x - a_i)(x - a_i)^T + c_i]^{-1}$	4	$[0, 10]^D$	-10	Shekel

framework of PSO. Note that the method can be theoretically used on most variants of PSO. For the proof of conceps, the method is described in this paper based on the traditional PSO algorithm [2].

In the PSO algorithm, the difference between two positions $pBest$ and $gBest$ and the current position is used as an estimate of the gradient. Then for each individual i at the tth iteration,

$$\begin{aligned} \Delta v_i^t = c_1 rand_1 \cdot (pBest_i - x_i^{t-1}) \\ + c_2 rand_2 \cdot (gBest - x_i^{t-1}) \end{aligned} \tag{8}$$

$$v_i^t = w_t v_i^{t-1} + \Delta v_i^t \tag{9}$$

where Δv_i^t is an estimation of the gradient; c_1 and c_2 are two positive real parameters; $rand_1$ and $rand_2$ are two D-dimensional random number vectors between $[0, 1]$; $pBest$ is the best position in the history of the individual; $gBest$ is the best position in the history of all individuals; v_i^t is updated along the Δv_i^t direction, and w_t denotes the intertia weight, which decreases as the search proceeds [2], [3].

Similar to Adam, the estimation of second-order momentum m_i^t should be the sum of squares of Δv_i^t. However, the gradient descent-based optimizer is designed for the optimization of neural network weights. The optimization of neural network weights is significantly different from the optimization problem that PSO is designed to solve, where the weights of the neural network are mostly concentrated in the $[0, 1]$ interval during initialization and remain concentrated in the $[0, 1]$ interval after optimization [8]. Similarly, its gradient has similar consistency on scale. Therefore, there will be no challenge on scale for the estimation of second-order momentum in Adam. However, in the optimization problems that PSO is designed to solve, taking the search behavior of PSO on the Sphere function [3] as an example, the scale is in the $[-100, 100]$ interval at initialization, and as the search converges, the solution gradually concentrates in the $[-10^{-10}, 10^{-10}]$ interval. At this point, the estimated terms of the second-order momentum have no more consistency in scale and are more likely to float up and down in extremes. Therefore, the proposed method adopts a normalization method similar to Batch Normalization [9] to ensure the scale consistency of the second-order momentum update.

$$m_i^t = \beta m_i^{t-1} + (1 - \beta)(\frac{\Delta v_i^t - mean(\Delta v_i^t)}{std(\Delta v_i^t)} + 1)^2 \tag{10}$$

where m_i^t is the estimation of individual i in for second-order momentum with initial value m_i^1 as $1_{D\times 1}$; β is the control parameter, and $mean(\Delta v_i^t)$ and $std(\Delta v_i^t)$ are the mean and standard deviation of velocity v_i^t. By this normalization, the velocity v_i^t is approximated by scaling it to a Gaussian distribution $\mathcal{N}(1, 1)$, consistent with the scale of m_i^t

And then, for each individual i, its position is updated by the velocity v_i^t and the dynamically adjusted learning rate.

$$x_i^t = x_{i-1}^t + \frac{\eta}{\sqrt{m_i^t + \epsilon}} \cdot v_i^t \tag{11}$$

The current velocity v_t is adaptively adjusted by $\frac{1}{\sqrt{m_t}}$, and ϵ is a small constant to prevent division by 0, usually taking the value 10^{-8} [5]; η is the learning rate control term. Similar to Adam, adding an explicit learning rate control term in the velocity adjustment term is more beneficial to control the convergence speed of the algorithm.a

2) Boundary Control: In this paper, the cases that exceed the search boundary are handled using a random re-initialization in the search space [3], i.e.

$$x_i^d = \begin{cases} x_i^d, & l_d \le x_i^d \le u_d \\ l_d + (\frac{rand}{4} \times (u_d - l_d)), & x_i^d < l_d \\ u_d - (\frac{rand}{4} \times (u_d - l_d)), & x_i^d > u_d \end{cases} \tag{12}$$

where u_d, l_d are the upper and lower bounds of the d-th dimension; $rand$ is a random number between $[0, 1]$.

C. Overall Algorithm

The complete AdamPSO algorithm is shown in Algorithm 1. For the initialization phase, the maximum velocity V_{max} is defined and the velocity v_i^1 is assigned with reference to [3]. In terms of implementation details, the intertia weight decreases as the search proceeds (Line 11); the absolute value of the velocity is truncated when it is greater than the maximum velocity V_{max} (Lines 22-26), and the rest of the execution is consistent with the descriptions in Section II-B1 and Section II-B2.

III. Experimental Results and Discussion

A. Benchmark Functions and Algorithm Configuration

The 12 test functions [7] used in this paper are listed in Table I. Where functions $f1$-$f5$ are unimodal functions; function $f6$ is step function; function $f7$ is noisy quartic function, and functions $f8$-$f11$, $f13$ are multimodal functions.

Parameters of AdamPSO and conventional PSO [2] are shown in Table II. Among them, the parameters common to the conventional PSO and AdamPSO are consistent with the conventionally recommended values [2], [10]. AdamPSO adds two parameters: the second-order momentum estimation update parameter β is set according to the convention in the Adam optimizer [5]; the learning rate η is taken as 1.2, and more discussion of the settings of this parameter is given in Section III-C.

The implementation of AdamPSO as well as the conventional PSO algorithm is partially referenced to [3].

TABLE II
Parameter Settings for PSO and AdamPSO.

Parameter	Value	Reference
Population size m	50	[10]
Intertia weight decrasing rate w_0, w_1	0.9, 0.5	[2]
PSO velocity update coefficient c_1, c_2	2, 2	[2]
Second order moment update coefficient β	0.001	[5]
learning rate η	1.2	N/A

TABLE III
Search Result Comparison
Between AdamPSO and Origin PSO [2].

Problem	PSO	AdamPSO
f_1	2.22e-18 (6.05e-18) −	6.54e-47 (2.32e-46)
f_2	6.88e-18 (8.88e-18) −	4.24e-24 (2.31e-23)
f_3	2.84e-02 (2.77e-02) −	1.48e-23 (3.32e-23)
f_4	7.69e-02 (7.39e-02) −	1.79e-14 (3.46e-14)
f_5	1.08e+01 (2.16e+01) ≈	9.69e+00 (2.21e+01)
f_6	9.57e-19 (1.53e-18) −	3.27e-31 (1.64e-30)
f_7	9.58e-03 (2.83e-03) −	1.55e-03 (6.43e-04)
f_8	-1.16e+04 (3.20e+02) ≈	-1.15e+04 (2.88e+02)
f_9	2.22e+01 (7.42e+00) +	3.26e+01 (9.35e+00)
f_{10}	3.29e-10 (3.53e-10) −	7.99e-15 (0.00e+00)
f_{11}	2.11e-02 (2.59e-02) ≈	1.18e-02 (1.51e-02)
f_{13}	-6.53e+00 (2.70e+00) ≈	-5.42e+00 (1.72e+00)
'+/−/≈'	1/7/4	

B. Comparison with PSO and Discussion

Following the best practices from [11], The Wilcoxon signed ranks test was used as a statistical tool to measure the performance of AdamPSO compared to PSO ($\alpha = 0.01$). Table III shows the mean and variance of PSO and AdamPSO on the twelve tested functions, and the group with the lowest mean is marked as gray.

From the experimental results, it can be observed that AdamPSO is significantly worse than PSO for only one function and significantly better than PSO for seven functions.

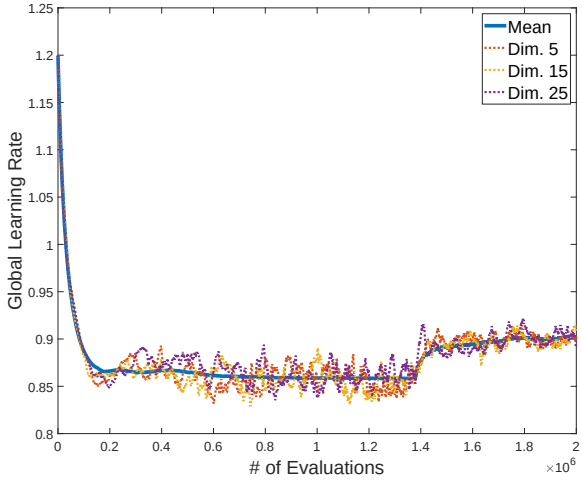

Fig. 1. Mean of estimated moment and its component from fifth, fifteenth and twenty-fifth dimensions in optimization process of f_1 Shere Model.

Specifically, AdamPSO performs significantly better in six out of seven unimodal functions. Intuitively, the smaller learning rate adjusted by the cumulative momentum in the late search period helps AdamPSO to enhance its local search capability. On the multimodal function, AdamPSO does not perform significantly differently from PSO. This may indicate that AdamPSO does not significantly enhance the global exploration ability early at the early stage of search, but it does not diminish the global exploration ability either.

C. Parameter Setting Tests

In this section, we discuss the effect of the learning rate parameter η on the performance of AdamPSO. The experiments were conducted by taking $\eta = 0.8, 1.0, 1.2, 1.4, 1.6$ and testing the performance of the algorithm with other parameters identically, and the results are shown in Table IV. In the table, all other parameters are compared with the recommended settings $\beta = 1.2$, quantifying by the Wilcoxon signed ranks test ($\alpha = 0.01$), and the best mean group is marked as gray.

It is observed that the lower learning rate groups ($\eta = 0.8, 1.0$) performs better in the unimodal functions, while the higher learning rate groups ($\eta = 1.0, 1.2$) performs better in the multimodal functions. Intuitively, lower learning rates are more helpful for local search , while higher learning rates may help the algorithm to jump out of the local optimum in more complex functions. Considering the logarithm of significant good and bad, and the robustness to complex functions, a recommended setting of $\eta = 1.2$ is given in this paper.

D. Adaptation Behavior Analysis

How the adaptive momentum estimation mechanism behaves in the search process is discussed in this section. Here, $\frac{\eta}{\sqrt{m_i^t}+\epsilon}$ is named the global learning rate. in Fig. 2 shows the average step size in the D dimension and its components in some dimensions along with the number of iterations. The

TABLE IV
PARAMETER SETTING TESTS FOR LEARNING RATE η .

Problem	$\eta = 0.8$	$\eta = 1$	$\eta = 1.4$	$\eta = 1.6$	$\boldsymbol{\eta = 1.2}$
f_1	1.78e-72 (9.73e-72) +	2.89e-78 (1.58e-77) +	8.43e-13 (1.44e-12) −	4.82e+02 (3.67e+02) −	6.54e-47 (2.32e-46)
f_2	3.08e-09 (1.06e-08) −	9.77e-15 (3.80e-14) −	4.16e-15 (5.68e-15) −	2.24e+00 (9.62e-01) −	4.24e-24 (2.31e-23)
f_3	4.67e-22 (1.53e-21) ≈	2.25e-30 (7.98e-30) +	5.92e-02 (4.32e-02) −	8.86e+03 (1.68e+03) −	1.48e-23 (3.32e-23)
f_4	9.83e-21 (1.87e-20) +	2.11e-19 (8.19e-19) +	2.85e-02 (2.36e-02) −	2.55e+01 (2.37e+00) −	1.79e-14 (3.46e-14)
f_5	3.17e+00 (3.09e+00) ≈	1.25e+01 (2.68e+01) ≈	3.09e+01 (3.19e+01) −	1.57e+03 (1.17e+03) −	9.69e+00 (2.21e+01)
f_6	9.35e-33 (9.66e-33) +	9.66e-33 (1.41e-32) +	6.11e-13 (7.19e-13) −	5.72e+02 (2.28e+02) −	3.27e-31 (1.64e-30)
f_7	3.53e-03 (2.02e-03) −	2.71e-03 (9.96e-04) −	8.52e-03 (2.46e-03) −	3.26e-01 (1.51e-01) −	1.55e-03 (6.43e-04)
f_8	-1.08e+04 (5.15e+02) −	-1.13e+04 (4.21e+02) −	-1.16e+04 (3.08e+02) ≈	-1.10e+04 (5.10e+02) −	-1.15e+04 (2.88e+02)
f_9	5.43e+01 (1.26e+01) −	4.50e+01 (9.63e+00) −	2.62e+01 (9.29e+00) +	1.49e+02 (2.75e+01) −	3.26e+01 (9.35e+00)
f_{10}	6.41e-01 (7.20e-01) −	8.94e-02 (3.40e-01) −	5.87e-07 (1.05e-06) −	6.62e+00 (1.54e+00) −	7.99e-15 (0.00e+00)
f_{11}	1.54e-02 (1.38e-02) ≈	1.56e-02 (1.84e-02) ≈	1.21e-02 (1.53e-02) ≈	2.99e+00 (1.52e+00) −	1.18e-02 (1.51e-02)
f_{13}	-7.13e+00 (3.19e+00) +	-6.04e+00 (2.91e+00) ≈	-5.02e+00 (8.03e-02) −	-4.68e+00 (2.51e-01) −	-5.42e+00 (1.72e+00)
'+/−/≈'	4/5/3	4/5/3	1/9/2	0/12/0	

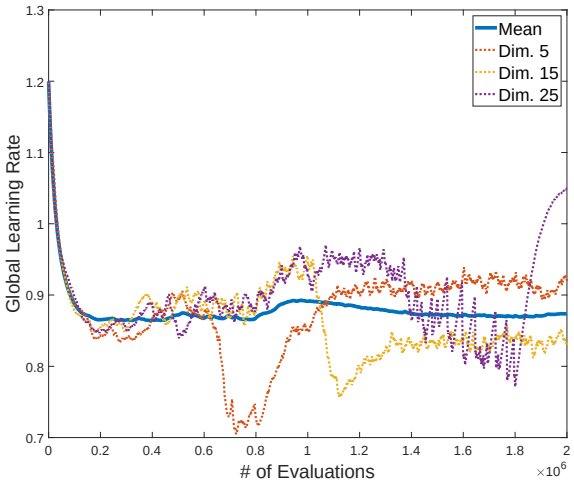

Fig. 2. Mean of estimated moment and its component from fifth, fifteenth and twenty-fifth dimensions in optimization process of f_5 Rosenbrock's Function.

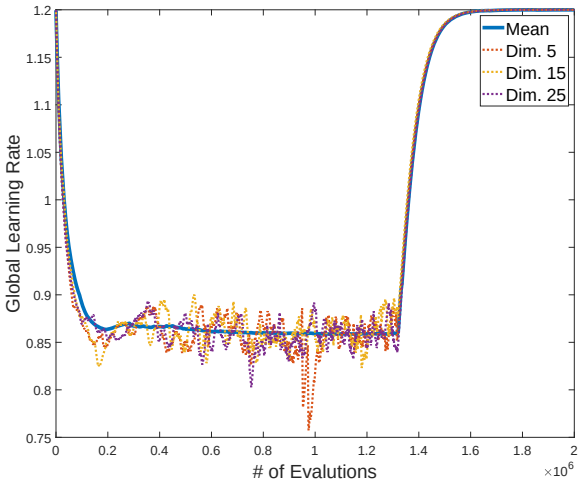

Fig. 4. Mean of estimated moment and its component from fifth, fifteenth and twenty-fifth dimensions in optimization process of f_2 Schwefel's Problem 2.22.

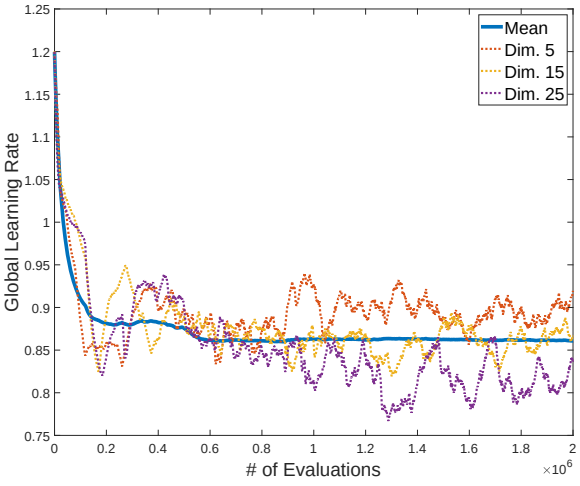

Fig. 3. Mean of estimated moment and its component from fifth, fifteenth and twenty-fifth dimensions in optimization process of f_8 Schwefel's Problem 2.26.

number of evaluations is adjusted to 10^6 for all problems for comparative fairness.

Fig. 1 shows the adaptive process on the f_1 Shere Model. This function is an unimodal function, so the landscapes of each dimension are identical, and the optimization process is synchronized, so the component global learning rate of the dimension floats in a small range around the mean.

Fig. 2 shows the adaptive process on f_5 Rosenbrock's Function. Although this function is an unimodal function, this function has the characteristic of inseparability, so that the landscapes of each dimension are very different. In this case, the optimization process of each dimension is not synchronized. As can be seen from the figure, the components of the dimension global learning rate change independently and are dynamically adjusted based on the historical momentum information of the dimension.

Fig. 3 shows the adaptive process on f_8 Schwefel's Problem 2.26. This function is a multimodal function. Although this function is also a separable function, in the multimodal case, the optimization process is more complex and the optimization

processes in each dimension are more likely to be unsynchronized. As can be seen in the figure, the component global learning rate of the dimension changes independently.

IV. Future Work

First, the proposed AdamPSO does not significantly improve the global search capability, although the local search capability is improved compared with the conventional PSO. Is it possible to improve the global search capability by improving some steps? For example, further research on the normalization mechanism, so that the learning rate can be adjusted to better fit the global search needs, is a direction for future work.

Second, in the experimental part, there are also some findings that are worthy of further exploration. First, it was observed that the global learning rate of all groups decreases rapidly to around 0.85 in the early stage and then fluctuates up and down. Is this an inevitable result brought about by the momentum estimation mechanism? Second, anomalous fluctuations similar to Fig. 4 occur during a certain optimization run of some functions. This occurs with high frequency on the function f_2, and with smaller probability on other functions. What is the reason for the appearance of this curve, which seems to be quite regular? Both of these questions deserve further exploration.

V. Conclusion

In this paper, we propose AdamPSO, which introduces the adaptive moment estimation method into the PSO framework and improves the learning rate mechanism of the PSO algorithm. With the cooperation of the newly added method, AdamPSO can adjust the learning rate adaptively for each dimension based on the historical momentum information. It is experimentally demonstrated that AdamPSO enhances the local search ability of PSO algorithm without weakening the global exploration ability of PSO algorithm. The ablation study of the learning rate parameter η proves the effectiveness of the parameter setting, while the analysis of the adaptive behavior brings more insight into the adaptive learning rate adjustment.

References

[1] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of IEEE International Conference on Neural Networks*, vol. 4, 1995, pp. 1942–1948.

[2] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proceedings of IEEE World Congress on Computational Intelligence*, 1998, pp. 69–73.

[3] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Transactions on Evolutionary computation*, vol. 10, no. 3, pp. 281–295, 2006.

[4] Y. Shi *et al.*, "Particle swarm optimization: developments, applications and resources," in *Proceedings of Congress on Evolutionary Computation*, vol. 1, 2001, pp. 81–86.

[5] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2015.

[6] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural Networks*, vol. 12, no. 1, pp. 145–151, 1999.

[7] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary computation*, vol. 3, no. 2, pp. 82–102, 1999.

[8] B. Li, B. Wu, J. Su, and G. Wang, "Eagleeye: Fast sub-net evaluation for efficient neural network pruning," in *Proceedings of European Conference on Computer Vision*, 2020, pp. 639–654.

[9] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of Conference on International Conference on Machine Learning*, 2015, p. 448–456.

[10] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *Proceedings of Congress on Evolutionary Computation*, vol. 3, 1999, pp. 1945–1950.

[11] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.