Research Article

Gaussian Quantum Bat Algorithm with Direction of Mean Best Position for Numerical Function Optimization

Xingwang Huang¹,¹ Chaopeng Li,² Yunming Pu,¹ and Bingyan He¹

¹Computer Engineering College, Jimei University, 185 Yinjiang Rd., Jimei District, Xiamen 361021, China

²Information Engineering College, Jimei University, 185 Yinjiang Rd., Jimei District, Xiamen 361021, China

Correspondence should be addressed to Xingwang Huang; huangxw@jmu.edu.cn

Received 19 September 2019; Revised 23 October 2019; Accepted 28 October 2019; Published 16 November 2019

Academic Editor: Rodolfo E. Haber

Copyright © 2019 Xingwang Huang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Quantum-behaved bat algorithm with mean best position directed (QMBA) is a novel variant of bat algorithm (BA) with good performance. However, the QMBA algorithm generates all stochastic coefficients with uniform probability distribution, which can only provide a relatively small search range, so it still faces a certain degree of premature convergence. In order to help bats escape from the local optimum, this article proposes a novel Gaussian quantum bat algorithm with mean best position directed (GQMBA), which applies Gaussian probability distribution to generate random number sequences. Applying Gaussian distribution instead of uniform distribution to generate random coefficients in GQMBA is an effective technique to promote the performance in avoiding premature convergence. In this article, the combination of QMBA and Gaussian probability distribution is applied to solve the numerical function optimization problem. Nineteen benchmark functions are employed and compared with other algorithms to evaluate the accuracy and performance of GQMBA. The experimental results show that, in most cases, the proposed GQMBA algorithm can provide better search performance.

1. Introduction

Recently, optimization problems are usually encountered in a mount of real-word areas such as artificial intelligence, computer science, pattern recognition, information theory, etc. Many of the actual optimization problems are frequently NP-hard problems, and searching for optimal solutions is pretty hard. It stands for the reason that it takes too long to solve these NP-hard optimization problems with traditional optimization methods. Therefore, different optimization techniques, especially bioinspired metaheuristic optimization algorithms or swarm intelligence (SI) optimization algorithms, have raised many researchers' growing interest in the past twenty years and they have proposed or developed various optimization algorithms such as particle swarm optimization (PSO) [1, 2], gravitational search algorithm (GSA) [3], ant colony optimization (ACO) [4], cuckoo search (CS) [5], and bat algorithm (BA) [6]. These algorithms are verified that they are very suited for optimizing problems such as

feature selection [7, 8], task scheduling [9], unit commitment [10], artificial neural networks [11], fuzzy control [12], parameter selection and optimization [13, 14], and numerical function optimization [15]. They have also been applied to multiobjective optimization problems [16]. Compared with traditional optimization techniques, these algorithms can provide better generalization ability and parallelism. Hence, in many high-dimensional optimization problems, these algorithms outperform the traditional optimization techniques.

The bat algorithm is a relatively new nature-inspired swarm-based optimization algorithm that was proposed by Yang in 2010 [6]. This algorithm mimics the foraging behavior of bats to search the optima and it has successfully combined the merits of many well-known algorithms in a structured way, such as PSO, simulated annealing (SA) [17] and genetic algorithm (GA) [18]. BA also inherits the simplicity of PSO and it has been proved to be more efficient than its predecessor PSO and GA, especially in low-dimensional cases. Also, it is easy to implement BO in various

computer languages. Hence, it has been applied to many engineering optimization problems [19]. However, due to its low population diversification, it may get trapped in local optima and premature convergence when solving high-dimensional optimization problems [20]. Therefore, to issue this deficiency, many bat algorithm variants are proposed to improve BA performance, such as CLBA [21], DLBA [22], IBA [23], HSBA [24], etc. QMBA [15] is also a variant of BA, which introduced the quantum behavior to improve the population diversity. Also, with the direction of mean best position in the later phase of searching, QMBA can convergence more quickly. QMBA is verified to be superior to the original BA and other four BA variants, including IBA, HSBA, MBA [25], and CBA [26].

However, all stochastic coefficients in the QMBA algorithm are generated using the uniform probability distribution, which can only provide a relatively small search range. Therefore, QMBA still faces a certain degree of premature convergence. To solve this issue, this article presents a technique of quantum-behaved BA directed by mean best position (QMBA) based on Gaussian distribution (GQMBA) for numerical function optimization. The employment of random Gaussian generation instead of random uniform generation in QMBA is an effective method to enhance the performance of QMBA in avoiding local solutions. In order to verify the performance of the Gaussian QMBA (GQMBA) technique, nineteen classical benchmark functions were employed, and the experimental results obtained by GQMBA over 30 trials were compared with some other algorithms mentioned in the literature.

The remainder of this article is structured as follows: the original bat algorithm is described in Sections 2 and 3 gives an overview to the QMBA. Section 4 presents the Gaussian quantum bat algorithm with the direction of mean best position. The following Section 5 provides the simulation results and comparison of this proposed technique. Finally, conclusions are made, and future research direction is presented in Section 6.

2. The Original BA Algorithm

Bat algorithm is a nature-inspired optimization algorithm inspired by foraging behavior of bats [6]. This technique is simple and easy to implement and efficient, which is swarm-based on a stochastic optimization method. When bats foraging, they search for prey and avoid obstacles by using the echolocation technique. The original BA employs a frequency-tuning approach to increase the diversification of the swarm, while, at the same time, it adopts the automatic zooming method to try to keep the balance of global search and local search during the search procedure by simulating the variations of pulse loudness and emission rates of bats when foraging. Based on three idealized rules [6], the foraging behavior of bats can be transformed to the bat algorithm. The following paragraphs present the details of BA.

In the original BA, each bat flies toward the prey, that is, moving toward the current global best position. The frequency vector (f_i), velocity vector (v_i), and position vector (x_i) of the artificial bats are updated during the process of iteration using the following equations:

$$f_i = f_{\min} + (f_{\max} - f_{\min})\beta, \quad (1)$$

$$v_i^t = v_i^{t-1} + (x_i^t - \text{gb}^t)f_i, \quad (2)$$

$$x_i^t = x_i^{t-1} + v_i^t, \quad (3)$$

where β is a uniform random number in the range of $[0, 1]$, $f_{\min}$ and $f_{\max}$ indicate the minimum and maximum frequency, respectively, gb^t means the current global best solution. With these equations, the global search capacity of BA can be guaranteed.

For the local search, to produce new solution for each bat when a solution is chosen from the current best solutions, a local random walk strategy is employed. This strategy can be described as follows:

$$x_{\text{new}} = x_{\text{old}} + \varepsilon \bar{A}^t, \quad (4)$$

where ε is a uniform random number in the range of $[-1, 1]$ and decides the direction of new solution. Here, A^t is the average loudness value of all bats at the t th iteration.

During the foraging process, bats will gradually adapt the values of loudness and pulse emission rate for the purpose of locating the prey. The loudness value A_i and pulse emission rate r_i can be updated in each cycle as follows:

$$A_i^{t+1} = \alpha A_i^t, \quad (5)$$

$$r_i^{t+1} = r_i^0 [1 - \exp(-\gamma t)], \quad (6)$$

where r^0 indicates the initial rate of pulse emission of i th artificial bat. α and γ are constants. The range of α is $[0, 1]$ and γ is a positive number ($\gamma > 0$). Actually, as the cooling coefficient in the SA, α decides the convergence of BA. For simplicity, $\alpha = \gamma$ is usually adopted in the researches.

The basic procedure of BA is described as the pseudo-code illustrated in Algorithm 1.

3. The QMBA Algorithm

The original bat algorithm has the characteristics of simplicity, easy to implement and quick convergence; hence, it has been applied to many optimization problems. However, BA performs bad in the multimodal cases, due to its low population diversity. Through the analysis of the trajectory of artificial bats, Zhu et al. [15] proposed the quantum-behaved bat algorithm with mean best position directed. In QMBA, the quantum-behaved mutation operator can increase the diversity of swam and it also can help to avoid premature convergence. Additionally, the mean best solution used in the later phase can quick up the convergence speed of the algorithm. The following paragraphs describe the details of QMBA [15].

QMBA is basically constructed on the basis of the original BA. The decreasing coefficient A and increasing coefficient r control the global search and local search, respectively. But the method to generate new candidate solutions is different from the original BA. The new method is described as follows:

```

Initialize the bat population  $x_i (i = 1, 2, \dots, n)$  and  $v_i$ ;
Define pulse frequency  $f_i$ , pulse rate  $r_i$  and the loudness  $A_i$ ;
while ( $t < t_{\max}$ ) do
    Generate new solutions by adjusting frequency, updating velocities and positions using equations (1)–(3);
    if ( $\text{rand} > r_i$ ) then
        Select a solution among the best solutions randomly;
        Generate a local solution around the selected best solution using equation (4);
    end if
    if ( $\text{rand} < A_i$  &  $f(x_i) < f(x_g)$ ) then
        Accept the new solutions;
        Increase  $r_i$  and reduce  $A_i$  using equations (5) and (6);
    end if
    Rank the bats and find the current best  $gb$ ;
     $t = t + 1$ ;
end while

```

ALGORITHM 1: Pseudocode of the BA algorithm.

$$x_{id}^t = \begin{cases} x_{id}^{t-1} + (gb_d - x_{id}^{t-1}) \times \eta, & \delta_d > TH, \\ x_{id}^{t-1} + \epsilon, & \delta_d \leq TH, \end{cases} \quad (7)$$

where η indicates a random number uniformly distributed in the range $[0, 1]$.

$$\delta_d = |gb_d - x_{id}^{t-1}|, \quad (8)$$

and it represents the distance between the d th dimension of current global best position gb in the swarm and the position of d th dimension of i th bat, rand is a uniform random number in $[0, 1]$. If the distance δ_d is smaller than the threshold TH , the i th bat can fly randomly. However, if the distance δ_d is larger than the threshold TH , then the i th bat flies toward to the current global best position.

For the local search, the random walk strategy is not employed again. According to certain mutation probability p_m , some of the bats will be mutated with quantum-behaved operator, which can be described as follows:

$$x_{id}^t = \begin{cases} gb_d^t + \mu \times |mbest_d - x_{id}^t| \times \ln\left(\frac{1}{U}\right), & \text{rand} < 0.5, \\ gb_d^t - \mu \times |mbest_d - x_{id}^t| \times \ln\left(\frac{1}{U}\right), & \text{rand} \geq 0.5, \end{cases} \quad (9)$$

where U is a random number in the range $[0, 1]$ generated by the uniform distribution and μ is a self-adaptive linear decreasing coefficient defined as

$$\mu = \mu_{\max} - \frac{(\mu_{\max} - \mu_{\min})}{t_{\max}} \times t, \quad (10)$$

and $\mu_{\max}$ and $\mu_{\min}$ are the initial and final values of μ . In the QMBA, $\mu_{\max} = 1$ and $\mu_{\min} = 0.5$ are adopted.

$$mbest = \frac{1}{M} \left(\sum_{i=1}^M P_{i1}^t, \sum_{i=1}^M P_{i2}^t, \dots, \sum_{i=1}^M P_{iD}^t \right), \quad (11)$$

where the $mbest$ denotes the mean best position, that is, the average value of P_i^t positions of all artificial bats. P_i^t represents the present best position of the i th bat, M indicates the size of swarm, and D represents the dimension of problem.

If a bat does not mutate with quantum-behaved operation mentioned above during the local search, then the position of the bat is updated as follows:

$$x_i^t = x_i^{t-1} + (mbest - x_i^{t-1})\phi, \quad (12)$$

where ϕ also denotes a random number of uniform distribution in the range $[0, 1]$.

The pseudocode of QMBA is presented in Algorithm 2.

4. The GQMBA Algorithm

Various novel variants of BA have been developed to improve the performance of the original BA in recent years. Most of these BA variants generate random numbers with uniform probability distribution. However, some researches have proved that other probability distributions, such as Gaussian (normal) probability distribution, can be a good choice to improve the performance of heuristic algorithms [27–29]. In fact, any long-tail distribution helps increase the step size and distance of the random walk.

In this section, following the same direction of research, we give out a combination of QMBA and Gaussian probability distribution, which is called Gaussian QMBA (GQMBA).

A random generation that the Gaussian probability distribution with a mean of 0 and a standard deviation of 1 is utilized for initializing stochastic coefficients of GQMBA. GQMBA offers a good trade-off between the probability of having numerous small amplitudes near the present position and the small probability of having a higher amplitude. This random generation allows bats fly away from the present position and jump out of local optima. It can not only promote the accuracy of the solutions but also improve the robustness of the optimization technique. As described in Section 3, there are three uniformly distributed random sequences in the search process of QMBA. Therefore, with the application of Gaussian random generation, the GQMBA algorithm can provide a wider search space and the performance of QMBA may be improved.

In this article, stochastic numbers in GQMBA are generated using the absolute value of Gaussian probability

```

Initialize the bat population  $x_i (i = 1, 2, \dots, n)$  and  $v_i$ ;
Define pulse frequency  $f_i$ , pulse rate  $r_i$  and the loudness  $A_i$ ;
while ( $t < t_{\max}$ ) do
  for  $i = 1$  to  $n$  do
    Generate new solutions by calculating the distance between the bat and current global best position, updating positions using
    equations (7) and (8);
    if ( $\text{rand} > r_i$ ) then
      if  $\text{rand} > p_m$  then
        Bats fly with quantum behavior and positions using equations (9)–(11);
      else
        The mean best position is used to guide other bats and position updated using equations (11) and (12);
      end if
    end if
    if ( $\text{rand} < A_i \ \&\& \ f(x_i) < f(\text{gb})$ ) then
      Accept the new solutions;
      Increase  $r_i$  and reduce  $A_i$  using equations (5) and (6);
    end if
    Rank the bats and find the current best gb;
  end for
   $t = t + 1$ ;
end while

```

ALGORITHM 2: Pseudocode of the QMBA algorithm.

```

Initialize the bat population  $x_i (i = 1, 2, \dots, n)$  and  $v_i$ ;
Define pulse frequency  $f_i$ , pulse rate  $r_i$  and the loudness  $A_i$ ;
while ( $t < t_{\max}$ ) do
  for  $i = 1$  to  $n$  do
    Generate new solutions by calculating the distance between the bat and current global best position, updating positions using
    equations (8) and (14);
    if ( $\text{rand} > r_i$ ) then
      if  $\text{rand} > p_m$  then
        Bats fly with quantum behavior and positions using equations (10), (11) and (15);
      else
        The mean best position is used to guide other bats and position updated using equations (11) and (16);
      end if
    end if
    if ( $\text{rand} < A_i \ \&\& \ f(x_i) < f(\text{gb})$ ) then
      Accept the new solutions;
      Increase  $r_i$  and reduce  $A_i$  using equations (5) and (6);
    end if
    Rank the bats and find the current best gb;
  end for
   $t = t + 1$ ;
end while

```

ALGORITHM 3: Pseudocode of the GQMBA algorithm.

distributions with a mean of zero and a standard deviation of one, that is, $\text{abs}(N(0, 1))$ or $|\text{randn}|$. The one-dimensional probability density function of $\text{abs}(N(0, 1))$ is defined by:

$$q(x) = \frac{2}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right), \quad x \geq 0. \quad (13)$$

The combination of QMBA and Gaussian probability distribution is simple but effective. Only three equations need to be modified. The three major highlights are described below.

Firstly, parameter η of equation (7) is modified according to the following equation:

$$x_{id}^t = \begin{cases} x_{id}^{t-1} + (\text{gb}_d - x_{id}^{t-1}) \times G, & \delta_d > \text{TH}, \\ x_{id}^{t-1} + \epsilon, & \delta_d \leq \text{TH}, \end{cases} \quad (14)$$

where $G = \text{abs}(N(0, 1))$.

Secondly, U of equation (9) is also replaced with the absolute value of the Gaussian probability distribution with a mean of zero and unit standard deviation. This quantum-behaved mutation operator now is updated according to

TABLE 1: F_1 – F_7 unimodal benchmark functions.

Function	D	Range	$f_{\min}$
$F_1(x) = \sum_{i=1}^n x_i^2$	30	$[-100, 100]$	0
$F_2(x) = \sum_{i=1}^n x_i + \prod_{i=1}^n x_i $	30	$[-10, 10]$	0
$F_3(x) = \sum_{i=1}^n (\sum_{j=1}^i x_j)^2$	30	$[-100, 100]$	0
$F_4(x) = \max_i \{ x_i , 1 \leq i \leq n\}$	30	$[-100, 100]$	0
$F_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	$[-30, 30]$	0
$F_6(x) = \sum_{i=1}^n ((x_i + 0.5))^2$	30	$[-100, 100]$	0
$F_7(x) = \sum_{i=1}^n ix_i^4 + \text{random}[0, 1]$	30	$[-1.28, 1.28]$	0

TABLE 2: F_8 – F_{13} multimodal benchmark functions.

Function	D	Range	$f_{\min}$
$F_8(x) = \sum_{i=1}^n -x_i \sin(\sqrt{ x_i })$	30	$[-500, 500]$	$-418.9829 \times D$
$F_9(x) = \sum_{i=1}^n [x_i^2 - 10 \cos(2\pi x_i) + 10]$	30	$[-5.12, 5.12]$	0
$F_{10}(x) = -20 \exp(-0.2 \sqrt{1/n \sum_{i=1}^n x_i^2}) - \exp(1/n \sum_{i=1}^n \cos(2\pi x_i)) + 20 + e$	30	$[-32, 32]$	0
$F_{11}(x) = (1/4000) \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos(x_i/\sqrt{i}) + 1$	30	$[-600, 600]$	0
$F_{12}(x) = (\pi/n) \{10 \sin(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 [1 + 10 \sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + \sum_{i=1}^n u(x_i, 10, 100, 4) + \sum_{i=1}^n u(x_i, 10, 100, 4)$ $y_i = 1 + (x_i + 1/4)$	30	$[-50, 50]$	0
$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m x_i & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m x_i & x_i < -a \end{cases}$			
$F_{13}(x) = 0.1 \{ \sin^2(3\pi x_1) + \sum_{i=1}^n (x_i - 1)^2 [1 + \sin^2(3\pi x_i + 1)] + (x_n - 1)^2 [1 + \sin^2(2\pi x_n)] \} + \sum_{i=1}^n u(x_i, 5, 100, 4)$	30	$[-50, 50]$	0

TABLE 3: F_{14} – F_{19} composite benchmark functions.

Function	D	Range	$f_{\min}$
F_{14} (CF1) $f_1, f_2, f_3, \dots, f_{10} = \text{Sphere Function}, [\sigma_1, \sigma_2, \sigma_3, \dots, \sigma_{10} = [1, 1, 1, \dots, 1]$ $[\lambda_1, \lambda_2, \lambda_3, \dots, \lambda_{10}] = [5/100, 5/100, 5/100, \dots, 5/100]$	30	$[-5, 5]$	0
F_{15} (CF2) $f_1, f_2, f_3, \dots, f_{10} = \text{Griewank's Function}, [\sigma_1, \sigma_2, \sigma_3, \dots, \sigma_{10} = [1, 1, 1, \dots, 1]$ $[\lambda_1, \lambda_2, \lambda_3, \dots, \lambda_{10}] = [5/100, 5/100, 5/100, \dots, 5/100]$	30	$[-5, 5]$	0
F_{16} (CF3) $f_1, f_2, f_3, \dots, f_{10} = \text{Griewank's Function}, [\sigma_1, \sigma_2, \sigma_3, \dots, \sigma_{10} = [1, 1, 1, \dots, 1]$ $[\lambda_1, \lambda_2, \lambda_3, \dots, \lambda_{10}] = [1, 1, 1, \dots, 1]$	30	$[-5, 5]$	0
F_{17} (CF4) $f_1, f_2 = \text{Ackley's Function}, f_3, f_4 = \text{Rastrigin's Function}$ $f_5, f_6 = \text{Weierstrass Function}, f_7, f_8 = \text{Griewank's Function}$ $f_9, f_{10} = \text{Sphere Function}, [\sigma_1, \sigma_2, \sigma_3, \dots, \sigma_{10} = [1, 1, 1, \dots, 1]$ $[\lambda_1, \lambda_2, \lambda_3, \dots, \lambda_{10}] = [5/32, 5/32, 5/32, \dots, 5/32]$	30	$[-5, 5]$	0
F_{18} (CF5) $f_1, f_2 = \text{Rastrigin's Function}, f_3, f_4 = \text{Weierstrass Function}$ $f_5, f_6 = \text{Griewank's Function}, f_7, f_8 = \text{Ackley's Function}$ $f_9, f_{10} = \text{Sphere Function}, [\sigma_1, \sigma_2, \sigma_3, \dots, \sigma_{10} = [1, 1, 1, \dots, 1]$ $[\lambda_1, \lambda_2, \lambda_3, \dots, \lambda_{10}] = [1/5, 1/5, 5/0.5, 5/0.5, 5/100, 5/100, 5/32, 5/32, 5/100, 5/100]$	30	$[-5, 5]$	0
F_{19} (CF6) $f_1, f_2 = \text{Rastrigin's Function}, f_3, f_4 = \text{Weierstrass Function}$ $f_5, f_6 = \text{Griewank's Function}, f_7, f_8 = \text{Ackley's Function}$ $f_9, f_{10} = \text{Sphere Function}$ $[\sigma_1, \sigma_2, \sigma_3, \dots, \sigma_{10} = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]$ $[\lambda_1, \lambda_2, \lambda_3, \dots, \lambda_{10}] = [0.1 * 1/5, 0.2 * 1/5, 0.3 * 5/0.5, 0.4 * 5/0.5, 0.5 * 5/100, 0.6 * 5/100, 0.7 * 5/32, 0.8 * 5/32, 0.9 * 5/100, 1 * 5/100]$	30	$[-5, 5]$	0

TABLE 4: The parameter settings of BA, QMBA, MFO, and GQMBA.

Algorithms	Parameter design
BA	$f_{\min} = 0, f_{\max} = 2, A = u(0, 1), r = 0.01, \alpha = 0.5, \gamma = 0.5$
QMBA	$A = u(0, 1), r = 0.01, \alpha = 0.5, \gamma = 0.5, \text{TH} = 0.005, P_m = 0.01$
MFO	Identical to the values in the original article
GQMBA	$A = u(0, 1), r = 0.01, \alpha = 0.5, \gamma = 0.5, \mu_{\max} = 0.9, \mu_{\min} = 0.5, \text{TH} = 0.005, p_m = 0.01$

$u(0, 1)$ denotes a uniform random number ranged in $[0, 1]$.

TABLE 5: Mean and standard deviations of the benchmark functions.

F	BA		QMBA		MFO		GQMBA	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
F_1	$3.25e+04$	$9.53e+03$	8.13e-04	2.87e-03	$1.67e+03$	$4.61e+03$	$1.37e+00$	$4.07e+00$
F_2	$2.64e+08$	$1.36e+09$	$4.70e-02$	$2.06e-01$	$3.27e+01$	$2.27e+01$	2.39e-02	3.77e-02
F_3	$8.15e+04$	$3.28e+04$	$7.99e+03$	$4.97e+03$	$1.61e+04$	$8.92e+03$	2.62e+03	1.43e+03
F_4	$6.45e+01$	$9.17e+00$	$1.65e+01$	$5.47e+00$	$5.77e+01$	$1.38e+01$	4.81e-01	7.13e-01
F_5	$6.67e+07$	$3.21e+07$	9.72e+01	$1.31e+02$	$2.69e+06$	$1.46e+07$	$1.09e+02$	1.30e+02
F_6	$3.14e+04$	$8.06e+03$	$1.85e+00$	$5.70e+00$	$1.66e+03$	$3.78e+03$	1.43e+00	3.63e+00
F_7	$6.18e+01$	$3.69e+01$	5.07e-02	2.41e-02	$2.12e+00$	$5.01e+00$	$5.14e-02$	$4.52e-02$
F_8	$-3.82e+03$	$1.41e+03$	$-6.66e+03$	$1.61e+03$	-8.70e+03	$8.74e+02$	$-6.99e+03$	7.49e+02
F_9	$3.64e+02$	$3.34e+01$	$4.25e+01$	$1.68e+01$	$1.47e+02$	$3.23e+01$	3.47e+01	1.19e+01
F_{10}	$1.99e+01$	$1.78e-01$	$2.06e+00$	$1.29e+00$	$1.30e+01$	$9.12e+00$	1.53e-02	4.69e-02
F_{11}	$2.83e+02$	$8.03e+01$	$1.71e-01$	$3.45e-01$	$1.51e+01$	$4.17e+01$	8.08e-02	2.31e-01
F_{12}	$1.01e+08$	$8.36e+07$	$5.93e-01$	$8.74e-01$	$2.51e-01$	$3.44e-01$	3.88e-02	8.50e-02
F_{13}	$2.38e+08$	$1.46e+08$	$5.46e-01$	$1.56e+00$	$4.49e-02$	1.64e-01	8.59e-02	$2.31e-01$
F_{14}	$1.22e+03$	$1.12e+02$	$4.06e+02$	$2.59e+02$	$5.08e+02$	1.76e+02	3.27e+02	$1.91e+02$
F_{15}	$1.22e+03$	$1.24e+02$	$4.98e+02$	$1.92e+02$	$6.35e+02$	$2.46e+02$	4.25e+02	1.83e+02
F_{16}	$1.59e+03$	$1.66e+02$	$1.25e+03$	$1.96e+02$	$1.02e+03$	$2.02e+02$	8.66e+02	1.60e+02
F_{17}	$1.44e+03$	$1.00e+02$	$1.10e+03$	$7.58e+01$	$1.16e+03$	$1.61e+02$	9.80e+02	1.46e+02
F_{18}	$1.45e+03$	$1.29e+02$	$7.44e+02$	$3.69e+02$	$5.83e+02$	$3.46e+02$	4.00e+02	2.61e+02
F_{19}	$1.39e+03$	$8.79e+01$	1.03e+03	4.32e+01	$1.18e+03$	$6.28e+01$	$1.04e+03$	$5.40e+01$

TABLE 6: p -values of the Wilcoxon rank-sum test over all runs.

F	GQMBA	MFO	QMBA	BA
F_1	0.0150	0.0176	N/A	$3.0199e-11$
F_2	N/A	$5.5727e-10$	0.0207	$3.0199e-11$
F_3	N/A	$8.8411e-07$	$3.8053e-07$	$3.0199e-11$
F_4	N/A	$3.0199e-11$	$3.0199e-11$	$3.0199e-11$
F_5	<i>0.4376</i>	$3.3679e-04$	N/A	$3.0199e-11$
F_6	N/A	0.3255	0.0133	$3.0199e-11$
F_7	<i>0.2707</i>	$5.5611e-04$	N/A	$3.0199e-11$
F_8	$7.1186e-09$	N/A	0.0168	$4.1997e-10$
F_9	N/A	$3.0199e-11$	<i>0.1188</i>	$3.0199e-11$
F_{10}	N/A	$2.3897e-08$	$7.7725e-09$	$1.4110e-09$
F_{11}	N/A	<i>0.4290</i>	0.0251	$3.0199e-11$
F_{12}	N/A	$3.5708e-06$	$4.0840e-05$	$3.0199e-11$
F_{13}	<i>0.0850</i>	N/A	$9.7917e-05$	$3.0199e-11$
F_{14}	N/A	$5.2640e-04$	<i>0.2340</i>	$3.0199e-11$
F_{15}	N/A	0.0023	<i>0.1055</i>	$3.0199e-11$
F_{16}	N/A	0.0021	$2.1947e-08$	$3.6897e-11$
F_{17}	N/A	$4.0840e-05$	$8.1200e-04$	$3.0199e-11$
F_{18}	N/A	0.0083	$1.0407e-04$	$3.3384e-11$
F_{19}	<i>0.3871</i>	$2.2273e-09$	N/A	$4.0772e-11$

N/A means not applicable. $p \geq 0.05$ have been italicized.

FIGURE 1: The average curve of fitness value for F_1 . FIGURE 2: The average curve of fitness value for F_2 .

$$x_{id}^t = \begin{cases} gb_d^t + \mu \times |mbest_d - x_{id}^t| \times \ln\left(\frac{1}{G}\right), & \text{rand} < 0.5, \\ gb_d^t - \mu \times |mbest_d - x_{id}^t| \times \ln\left(\frac{1}{G}\right), & \text{rand} \geq 0.5, \end{cases} \quad (15)$$

where $G = \text{abs}(N(0, 1))$. Note that, according to equation (13), $q(0) = 0$; therefore, $\text{abs}(N(0, 1))$ satisfies the domain requirement of logarithmic function (> 0).

Thirdly, the random number ϕ in equation (12) is modified according to the following equation:

$$x_i^t = x_i^{t-1} + (mbest - x_i^{t-1})G, \quad (16)$$

FIGURE 3: The average curve of fitness value for F_3 . FIGURE 4: The average curve of fitness value for F_4 .

where $G = \text{abs}(N(0, 1))$.

Overall, in GQMBA, the present global best solution guides the exploration phase to guarantee convergence, while the Gaussian quantum-behaved mutation operator and mean

best position contribute to the exploitation phase to escape from local optimum and prevent premature convergence.

Based on the above description, the pseudocode of the GQMBA algorithm is summarized in Algorithm 3.

FIGURE 5: The average curve of fitness value for F_5 . FIGURE 6: The average curve of fitness value for F_6 .

5. Experiments and Discussion

In this section, nineteen classical benchmark functions are illustrated in Tables 1–3, which are adopted to test the performance of the GQMBA algorithm. These benchmark functions are usually employed in numerical optimization

techniques [15, 30–32]. In this article, the nineteen benchmark functions are grouped into three categories. The first category includes seven unimodal functions, which have only one optimal solution and are efficient to verify metaheuristic optimization techniques in terms of convergence speed and exploitation capability. The second category

FIGURE 7: The average curve of fitness value for F_7 . FIGURE 8: The average curve of fitness value for F_8 .

includes six multimodal functions, which have an exponential increasing number of local minima. Therefore, these multimodal functions are suitable for examining the local solutions avoidance and exploration capability of algorithms. The third category includes six composite functions, which are very complex with the combination of different rotated, shifted, and biased multimodal test functions. These

composite functions are highly similar to the actual applications and suitable to benchmark the performance of methods in terms of balanced global search and local search. In these tables, D denotes the dimension of the solution space, Range is the boundary of the search space, and the global best value f_m is also given in column 4. Thirty independent tests are completed for every benchmark

FIGURE 9: The average curve of fitness value for F_9 . FIGURE 10: The average curve of fitness value for F_{10} .

function. All the tests illustrated in this research are performed on a PC with Intel (R) Core (TM) i5-6500 3.20 GHz CPU and 8.0 GB RAM of memory, and the codes are implemented in Matlab 2014a.

Since metaheuristic algorithms belong to stochastic optimization methods, they need to be completed at least over 10 independent runs for producing meaningful statistical consequences. Besides the mean and standard

deviation, statistical tests, such as Wilcoxon rank-sum test, should be conducted to verify the significance of the results based on every independent runs. In this article, the non-parametric Wilcoxon rank-sum tests are completed to verify whether there exists a statistical difference between the results obtained by GQMBA and the results searched by the other algorithms. A p value of less than 0.05 (< 0.05) means that there exists statistical difference between the

FIGURE 11: The average curve of fitness value for F_{11} . FIGURE 12: The average curve of fitness value for F_{12} .

performances of the two algorithms, while a p value of greater than 0.05 (≥ 0.05) denotes that the performances are statistically similar.

Considering that QMBA has been verified to be more efficient than other variants of BA [15], therefore, GQMBA is compared with the original BA and QMBA as well as a new metaheuristic algorithm MFO [30] to verify its efficiency.

MFO is a novel nature-inspired heuristic algorithm. It shows high and competitive global search ability in multimodal functions and local search ability in unimodal functions. Also, MFO can balance global search and local search properly. Comparing with PSO, GSA, BA, FPA, SMS, FA, and GA, it can provide promising and competitive performance [30]. So MFO is selected as the comparative

FIGURE 13: The average curve of fitness value for F_{13} . FIGURE 14: The average curve of fitness value for F_{14} .

algorithm. In all swarm-based algorithms mentioned above, the maximum number of iterations is taken as 1000 for unimodal and multimodal functions, while the maximum number of iterations is taken as 100 for composite functions due to its high complexity, and the size of population is set as 50. Table 4 shows the other parameter settings for each algorithm. As presented in Tables 1–3, nineteen classical

benchmark functions are employed. The experimental results are presented in Tables 5 and 6 and Figures 1–19. Note that the optimal mean (Mean) and the optimal standard deviations (SD) of the results obtained by the four methods for each function are illustrated in bold.

As can be seen from Table 5, the GQMBA provides the best performance on most test functions, followed by the

FIGURE 15: The average curve of fitness value for F_{15} . FIGURE 16: The average curve of fitness value for F_{16} .

QMBA and MFO algorithms. And they are much better than the BA algorithm. To some extent, the results obtained also demonstrate that BA is easy to get trapped into local minimums when the dimension of search space is high.

The GQMBA algorithm obtains the best mean on 14 benchmark functions of 19 test functions, except F_1 , F_5 , F_7 , F_8 , and F_{19} . At the same time, GQMBA provides the most stable solutions on fourteen benchmark functions, except F_1 ,

F_7 , F_{13} , F_{14} , and F_{19} . The p -values in Table 6 illustrate that the superiority of the GQMBA algorithm is statistically significant on 9 benchmark functions, including F_2 , F_3 , F_4 , F_{10} , F_{12} , F_{16} , F_{17} , and F_{18} , which cover the unimodal, multimodal, and composite benchmark functions. On the other functions, GQMBA and QMBA perform statistically similar and there is no significant difference between these two algorithms, except F_1 and F_8 . With the analysis above, it

FIGURE 17: The average curve of fitness value for F_{17} . FIGURE 18: The average curve of fitness value for F_{18} .

can be concluded that the introduction of Gaussian probability distribution to QMBA is an effective mechanism and GQMBA has the significant advantage over the other three algorithms in terms of accuracy, stability, and local minimum avoidance.

Figures 1–19 demonstrate the average curves of fitness values obtained by four algorithms on nineteen functions. The values presented in these curves are the average

function fitness values obtained from 30 independent tests. These figures show that the original BA and MFO converge quickly with few iterations, but they are easy to get trapped into the local optimum in many cases. These figures also demonstrate that the convergence speed of GQMBA is similar to QMBA, while GQMBA can provide better accuracy and prevent premature convergence on most benchmark functions.

FIGURE 19: The average curve of fitness value for F_{19} .

TABLE 7: Average computational time of approaches on each benchmark function.

f	BA	MFO	QMBA	GQMBA
f_1	1.10036	0.469359	0.889615	0.929227
f_2	1.04243	0.524513	0.921171	0.980138
f_3	3.0765	2.43098	2.88777	2.95636
f_4	1.07779	0.567324	1.00172	1.03555
f_5	1.22965	0.71997	1.14489	1.19135
f_6	1.09087	0.599222	1.04263	1.08032
f_7	1.35753	0.886334	1.29639	1.33972
f_8	1.15973	0.654205	1.10923	1.14526
f_9	1.18425	0.672098	1.11233	1.16105
f_{10}	1.23347	0.716191	1.16477	1.19911
f_{11}	1.29489	0.754089	1.20586	1.25417
f_{12}	1.99625	1.43408	1.90856	1.95945
f_{13}	1.98464	1.42408	1.89719	1.93967
f_{14}	18.8752	16.5838	22.6735	17.1589
f_{15}	18.325	18.08	18.72	18.7628
f_{16}	19.9876	25.5575	23.0384	21.1629
f_{17}	27.0739	24.9261	23.0546	28.294
f_{18}	22.5187	22.7105	22.3053	22.7562
f_{19}	22.4076	21.9222	22.2938	22.2832

In order to compare the performance in terms of computational time, the average computational time of approaches on each test function over 30 independent runs is also provided in Table 7. As can be seen from this table, for unimodal and multimodal benchmark functions, the performance of MFO in terms of computational time is best and the average computational time of GQMBA is slightly more than that of QMBA. For composite benchmark functions, the computational time of GQMBA is similar to other approaches. The obtained results confirm that GQMBA can achieve better results with slightly more computational time.

6. Conclusions and Future Research Directions

In this article, novel Gaussian quantum bat algorithm with direction of mean best position (GQMBA) is proposed to simultaneously promote the search accuracy and stability of the original BA as well as QMBA. QMBA is combined with Gaussian probability distribution, which can improve the diversity, magnify the search range, and avoid falling into local optimum. Besides, GQMBA also inherits the characteristics of the original BA and QMBA, including simplicity, feasibility, and ease to implement. Nineteen benchmark

functions are experimented and results show that GQMBA outperforms other algorithms. In summary, the proposed GQMBA is efficient, and it is a good alternative method to solve the numerical function optimization.

However, there still exist some deficiencies in GQMBA. One of the major deficiencies is that it has plenty of parameters to be set. Few works have already been done to reduce or make BA without any parameter as presented in [33, 34]. Therefore, in the future, we could also do some works to reduce the number of parameters in order to make GQMBA simpler but still efficient.

Finally, the proposed GQMBA algorithm might be a good choice to integrate with angle modulation method [35–37] to solve binary optimization problems. We are suggesting angle modulation method because it can change the search space from high dimension to low dimension. Hence, it can improve the accuracy and reduce the search time. In addition, the GQMBA algorithm will be applied to more optimization applications such as task scheduling, image segmentation, logic circuit design, etc.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the Scientific Research Foundation of Jimei University, China, under Grant ZQ2019006 and in part by the Fujian Provincial Science and Technology Program Project under Grants 2019H0021 and 2017H0026.

References

- [1] K. James and E. Russell, "Particle swarm optimization," in *Proceedings of the 1995 IEEE International Conference on Neural Networks*, pp. 1942–1948, Perth, Western Australia, December 1995.
- [2] Y. Ning, Z. Peng, Y. Dai, D. Bi, and J. Wang, "Enhanced particle swarm optimization with multi-swarm and multi-velocity for optimizing high-dimensional problems," *Applied Intelligence*, vol. 49, no. 2, pp. 335–351, 2019.
- [3] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, "Gsa: a gravitational search algorithm," *Information Sciences*, vol. 179, no. 13, pp. 2232–2248, 2009.
- [4] M. Dorigo and G. Di Caro, "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, Washington, DC, USA, July 1999.
- [5] A. H. Gandomi, X.-S. Yang, and A. H. Alavi, "Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems," *Engineering with Computers*, vol. 29, no. 1, pp. 17–35, 2013.
- [6] X. S. Yang, "A new metaheuristic bat-inspired algorithm," in *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, pp. 65–74, Springer, Berlin, Germany, 2010.
- [7] M. Mafarja and S. Mirjalili, "Whale optimization approaches for wrapper feature selection," *Applied Soft Computing*, vol. 62, pp. 441–453, 2018.
- [8] L. Zhang, L. Shan, and J. Wang, "Optimal feature selection using distance-based discrete firefly algorithm with mutual information criterion," *Neural Computing and Applications*, vol. 28, no. 9, pp. 2795–2808, 2017.
- [9] B. Jana, M. Chakraborty, and T. Mandal, "A task scheduling technique based on particle swarm optimization algorithm in cloud environment," in *Soft Computing: Theories and Applications*, pp. 525–536, Springer, Berlin, Germany, 2019.
- [10] R. K. Srikanth, L. Panwar, B. K. Panigrahi, and R. Kumar, "Binary whale optimization algorithm: a new metaheuristic approach for profit-based unit commitment problems in competitive electricity markets," *Engineering Optimization*, vol. 51, no. 3, pp. 369–389, 2019.
- [11] D. J. Armaghani, M. Hasanipanah, A. Mahdiyar, M. Z. A. Majid, H. B. Amnieh, and M. M. Tahir, "Airblast prediction through a hybrid genetic algorithm-ann model," *Neural Computing and Applications*, vol. 29, no. 9, pp. 619–629, 2018.
- [12] R.-E. Precup and R.-C. David, *Nature-Inspired Optimization Algorithms for Fuzzy Controlled Servo Systems*, Butterworth-Heinemann, Oxford, UK, 2019.
- [13] I. la Fé-Perdomo, G. Beruvides, R. Quiza, R. Haber, and M. Rivas, "Automatic selection of optimal parameters based on simple soft-computing methods: a case study of micro-milling processes," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 2, pp. 800–811, 2019.
- [14] A. Tharwat and A. E. Hassanien, "Quantum-behaved particle swarm optimization for parameter optimization of support vector machine," *Journal of Classification*, pp. 1–23, 2019.
- [15] B. Zhu, W. Zhu, Z. Liu, Q. Duan, and L. Cao, "A novel quantum-behaved bat algorithm with mean best position directed for numerical optimization," *Computational Intelligence and Neuroscience*, vol. 2016, Article ID 6097484, 17 pages, 2016.
- [16] G. Beruvides, R. Quiza, and R. E. Haber, "Multi-objective optimization based on an improved cross-entropy method. a case study of a micro-scale manufacturing process," *Information Sciences*, vol. 334–335, pp. 161–173, 2016.
- [17] P. J. Van Laarhoven and E. H. Aarts, "Simulated annealing," in *Simulated Annealing: Theory and Applications*, pp. 7–15, Springer, Berlin, Germany, 1987.
- [18] S. Mirjalili, "Genetic algorithm," in *Evolutionary Algorithms and Neural Networks*, pp. 43–55, Springer, Berlin, Germany, 2019.
- [19] T. Jayabarathi, T. Raghunathan, and A. Gandomi, "The bat algorithm, variants and some practical engineering applications: a review," in *Nature-Inspired Algorithms and Applied Optimization*, pp. 313–330, Springer, Berlin, Germany, 2018.
- [20] A. R. Jordehi, "Chaotic bat swarm optimisation (cbso)," *Applied Soft Computing*, vol. 26, pp. 523–530, 2015.
- [21] J.-H. Lin, C.-W. Chou, C.-H. Yang, and H.-L. Tsai, "A chaotic levy flight bat algorithm for parameter estimation in nonlinear dynamic biological systems," *Computer and Information Technology*, vol. 2, no. 2, pp. 56–63, 2012.
- [22] J. Xie, Y. Zhou, and H. Chen, "A novel bat algorithm based on differential operator and levy flights trajectory," *Computational Intelligence and Neuroscience*, vol. 2013, Article ID 453812, 13 pages, 2013.
- [23] S. Yilmaz and E. U. Kucuksille, "Improved bat algorithm (iba) on continuous optimization problems," *Lecture Notes on Software Engineering*, vol. 1, no. 3, pp. 279–283, 2013.

- [24] G. Wang and L. Guo, "A novel hybrid bat algorithm with harmony search for global numerical optimization," *Journal of Applied Mathematics*, vol. 2013, Article ID 696491, 21 pages, 2013.
- [25] S. Yilmaz, E. U. Kucuksille, and Y. Cengiz, "Modified bat algorithm," *Elektronika Ir Elektrotechnika*, vol. 20, no. 2, pp. 71–78, 2014.
- [26] H. Afrabandpey, M. Ghaffari, A. Mirzaei, and M. Safayani, "A novel bat algorithm based on chaos for optimization tasks," in *Proceedings of the 2014 Iranian Conference on Intelligent Systems (ICIS)*, pp. 1–6, IEEE, Bam, Iran, February 2014.
- [27] X. Cai, L. Wang, Q. Kang, and Q. Wu, "Bat algorithm with Gaussian walk," *International Journal of Bio-Inspired Computation*, vol. 6, no. 3, pp. 166–174, 2014.
- [28] L. dos Santos Coelho, "Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems," *Expert Systems with Applications*, vol. 37, no. 2, pp. 1676–1683, 2010.
- [29] J. Liu, W. Xu, and J. Sun, "Quantum-behaved particle swarm optimization with mutation operator," in *Proceedings of the 17th IEEE international conference on tools with artificial intelligence (ICTAI'05)*, p. 4, IEEE, Hong Kong, China, November 2005.
- [30] S. Mirjalili, "Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm," *Knowledge-Based Systems*, vol. 89, pp. 228–249, 2015.
- [31] K. Chen, F. Zhou, and A. Liu, "Chaotic dynamic weight particle swarm optimization for numerical function optimization," *Knowledge-Based Systems*, vol. 139, pp. 23–40, 2018.
- [32] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Advances in Engineering Software*, vol. 95, pp. 51–67, 2016.
- [33] I. Fister Jr., I. Fister, and X. S. Yang, "Towards the development of a parameter-free bat algorithm," in *Proceedings of the 2015 2nd Student Computer Science Research Conference (StuCoSReC)*, pp. 31–34, Ljubljana, Slovenia, October 2015.
- [34] I. Fister, U. Mlakar, and X. S. Yang, "Parameterless bat algorithm and its performance study," in *Nature-Inspired Computation in Engineering*, pp. 267–276, Springer, Berlin, Germany, 2016.
- [35] B. J. Leonard, A. P. Engelbrecht, and C. W. Cleghorn, "Critical considerations on angle modulated particle swarm optimisers," *Swarm Intelligence*, vol. 9, no. 4, pp. 291–314, 2015.
- [36] G. Pampara, *Angle modulated population based algorithms to solve binary problems*, Ph.D. thesis, University of Pretoria, Pretoria, South Africa, 2013.
- [37] X. Huang, P. Li, and Y. Pu, "Amplitude angle modulated bat algorithm with application to zero-one knapsack problem," *IEEE Access*, vol. 7, pp. 27957–27969, 2019.