Front. Comput. Sci.
DOI 10.1007/s11704-016-5556-9

RESEARCH ARTICLE

Sequential quadratic programming enhanced backtracking search algorithm

Wenting ZHAO[1], Lijin WANG[1,2], Yilong YIN (✉)[1], Bingqing WANG[1], Yuchun TANG[3]

1 School of Computer Science and Technology, Shandong University, Jinan 250101, China
2 College of Computer and Information Science, Fujian Agriculture and Forestry University, Fuzhou 350002, China
3 Research Center for Sectional and Imaging Anatomy, Shandong University School of Medicine, Jinan 250012, China

Abstract In this paper, we propose a new hybrid method called SQPBSA which combines backtracking search optimization algorithm (BSA) and sequential quadratic programming (SQP). BSA, as an exploration search engine, gives a good direction to the global optimal region, while SQP is used as a local search technique to exploit the optimal solution. The experiments are carried on two suits of 28 functions proposed in the CEC-2013 competitions to verify the performance of SQPBSA. The results indicate the proposed method is effective and competitive.

Keywords numerical optimization, backtracking search algorithm, sequential quadratic programming, local search

1 Introduction

Optimization algorithm plays an important role in applied mathematics, decision sciences, and physical analysis aiming to obtain the minimum or maximum of the objective function. In general, the optimization algorithms can be divided into stochastic optimization algorithms and deterministic optimization algorithms based on whether there is randomness or uncertainty during the optimization process.

Broadly speaking, the natured-inspired approach is one kind of stochastic optimization algorithm. Researchers have come up with plenty of nature-inspired models to design evolutionary algorithm. Genetic algorithm (GA) proposed in [1] explores the optima by simulating the biological evolution process based on genes and chromosomes in nature. In differential evolutionary algorithm (DE) [2], the mutation and crossover operators are performed on the basis of differences between parent individuals. Ant colony optimization algorithm (ACO) [3] is inspired by ants foraging process. Particle swarm optimization algorithm (PSO) [4,5] models the behavior of individual exploration and group cooperation in birds foraging to locate the optima. Moreover, some improved and promising variants are also proposed. For example, Chen et al. proposed an improved particle swarm optimization with an aging leader and challengers (ALC-PSO) [6]. Yu et al. presented an enhanced differential evolution with two-level parameter adaptation by designing a new mutation strategy inspired the greedy DE/best/1 strategy [7].

BSA is a novel population based stochastic algorithm used for solving continuous numerical optimization that was first proposed by Civicioglu [8]. Similar to general evolutionary algorithm, BSA consists of the following steps: initialization, mutation, crossover, and selection. The procedure of BSA generating trial individuals using new designed mutation and crossover operators ensures its strong capability to solve optimization problems. BSA is controlled by a single parameter and not sensitive to the initial solution. Since the previous generation population information is used to guide the evolutionary process towards the optimal solution, BSA has shown promising performance for functions proposed in CEC-2005 and CEC-2011 competition [8].

Due to its promising performance, BSA has been ap-

Received December 24, 2015; accepted September 29, 2016

E-mail: yilong.ylyin@sdu.edu.cn

plied to different kinds of engineering optimization problems recently. A neural classifier optimized using BSA was presented by Agarwal et al. in [9] which has been proved to exhibit better results. In order to improve the effectiveness of the fault measurement method, Yang et al. adopted BSA to design the optimal chaotic excitation [10]. Zhang et al. combined BSA with three constraint handling methods to improve search efficiency for constrained optimization problems in [11]. Zhao et al. adopted BSA with DE and IBGA methods at different evolutionary stages and verified BSA's excellent robustness for complex constrained problems [12]. For complementary metal oxide semiconductor (CMOS) problem, Mallick et al. proposed BSA-DE by combining differential evolution algorithm in the mutation step [13]. Utilizing experience may make BSA converge slowly and prejudice exploitation on later iteration stage. Wang et al. proposed a hybrid algorithm HBD [14] by using differential evolution to enhance exploitation ability of BSA. A memetic BSA named MBSOA [15] was proposed by Ali et al. In MBSOA, random walk with direction exploitation method is combined with BSA as a local search engine to refine the best obtain solution at each iteration and speed up the procedure for solving economic dispatch problem. According to [13–16], the performance of BSA can be further improved by combining a local search method. In the light of this, we intend to seek a local search technique with strong search ability to enhance the performance of BSA.

The sequential quadratic programming (SQP) method, a gradient-based deterministic method, decomposes a complex nonlinear problem into a series of quadratic programming problems which are easy to solve. SQP uses derivative information when solving quadratic programming problems. As a result, SQP exhibits rapid decreasing speed. However, SQP is easy to get stuck in the local optima, thus, it can be regarded as another local search technique. Recently, SQP has been invoked into [17–20], and shown promising performance. Moreover, researches have paid more and more attention combining different search optimization algorithms or machine learning methods to improve the performance for real-world optimization problems, e.g. OLPSO [21]. Some good surveys about hybrid meta-heuristics or machine learning methods can be found in [22–24].

In the light of the above, we also concern on a hybrid meta-heuristic algorithm, called SQPBSA, which combines BSA and SQP. SQP is used in the earlier stage of the evolutionary of BSA to improve convergence speed and to favor exploitation. We use 28 benchmark functions to verify the performance of SQPBSA, and the results show improvement in effectiveness and efficiency of hybridization of BSA and SQP. The major advantages of our approach are as follows: (i) SQP helps SQPBSA converge fast, and keep the balance between exploration and exploitation. (ii) SQP is embedded in BSA as a component, therefore, SQPBSA does not destroy the structure of BSA, and it is still very simple.

The remainder of this paper is organized as follows. Section 2 introduces backtracking search algorithm and sequential quadratic programming. Section 3 gives description of the proposed method. Results are presented in Section 4 and the discussions are made in Section 5. Section 6 concludes this paper.

2 Preliminaries

In this section, we first introduce the general form of optimization problem, and then we describe BSA and SQP.

2.1 Problem formulation

Generally speaking, optimization problem can be presented as the following form.

$$\begin{aligned} &\text{Minimize} && f(x), \\ &\text{Subject to} && g_i(x) = 0, \quad i = 1, 2, \ldots, m_e; \\ & && g_j(x) \leqslant 0, \quad j = m_e + 1, m_e + 2, \ldots, m, \end{aligned} \tag{1}$$

where $f(x)$ is the objective function subject to constraints $g(x)$, m_e and m are numbers of equality and total constraints respectively. The first step of solving problem is to determine the objective and constraint functions. Then, a suitable method is employed to find the optimal vector x that is supposed to minimize the objective function and satisfy the equality and inequality constraints.

2.2 The basic idea of BSA

The backtracking search optimization algorithm is a new population-based stochastic search method [8]. BSA is capable of solving multimodal problems with a simple structure. BSA processes a memory to store previous generations which can guide the population towards global optimal taking advantages from historical experiences. To implement BSA, the following processes need to be performed.

BSA maintains the population P and historical population $oldP$ during evolutionary process. Each population consists of N individuals, and each individual keeps D genes. At the tth generation, D genes of the ith individual in population P or $oldP$ can be represented as $p_i^t = (p_{i,1}^t, p_{i,2}^t, \ldots, p_{i,D}^t)$. BSA initializes P and $oldP$ with Eqs. (2) and (3) respectively where $i = 1, 2, \ldots, N$, $j = 1, 2, \ldots, D$, low_j and up_j is denoted as the lowest and highest boundary constraint of the jth dimension in each individual, and U is the uniform distribution.

$$p_i \sim U(low_j, up_j). \tag{2}$$

$$oldp_i \sim U(low_j, up_j). \tag{3}$$

$OldP$ is constantly updated at the start of each iteration according to Eqs. (4) and (5) where $a, b \sim U(0, 1)$. $oldP$ produces a population from a random selection of former generation of P according to Eq. (4). Then, Eq. (5) reorders the selected population P as the current historical population $oldP$. In the following step, $oldP$ is involved in the calculation of search direction.

$$oldp_i = \begin{cases} p_i, & a < b; \\ oldp_i, & \text{otherwise.} \end{cases} \tag{4}$$

$$oldp_i := permuting(oldp_i). \tag{5}$$

BSA generates trial vectors through mutation and crossover operators which can be presented in Eqs. (6) and (7). Firstly, BSA has a random mutation strategy that uses only one direction individual $oldp_i$ for each target individual p_i. BSA generates a trial population, taking advantage of its experiences from previous generations. F controls the amplitude of the search-direction matrix. The initial form of the trial individual m_i is created by Eq. (6).

$$m_i = p_i + F \times (oldp_i - p_i). \tag{6}$$

The trial individual V is finally obtained by Eq.(7). BSA generates a binary integer-valued matrix called map to guide crossover directions. Trial individuals with better fitness values for the optimization problem are evolved into the target population individuals. Equation (7) shows BSA's crossover strategy.

$$v_{i,j} = \begin{cases} p_{i,j}, & map_{i,j} = 1; \\ m_{i,j}, & \text{otherwise.} \end{cases} \tag{7}$$

In the following stage, the target individual P_i is replaced by the trail vector V_i which has better fitness values than the corresponding P_i according to a greedy selection mechanism shown in Eq. (8).

$$p_i^{t+1} = \begin{cases} v_i^t, & f(v_i^t) \leqslant f(p_i^t); \\ p_i^t, & \text{otherwise.} \end{cases} \tag{8}$$

According the above description, the procedure of BSA can be summarized in Algorithm 1.

2.3 The basic idea of SQP

The SQP method is an iterative gradient-based method for solving nonlinear numerical optimization problems. In SQP, the original problem is converted into the corresponding quadratic programming sub-problems by Lagrange-Newton method using the gradient of objective and constraint functions. The search direction is obtained from quadratic programming, and the step length is calculated by minimizing the merit function. Both of them are used to form a new iterate. The process can be referred to [25]. This algorithm was first proposed by Wilson [26] and it is still considered an efficient way after the past two decades. Based on the characteristics of the gradient descent, it has fast convergence speed and high efficiency. We take the formulation of SQP subroutine from [17].

Algorithm 1 Backtracking search algorithm

Step 1: initiate the population P and the historical population $oldP$ containing N individuals randomly from search space.
Step 2: **while** the stop condition is not satisfied **do**
Step 3: selection-I: $oldP = P$ in the case of $a < b$, where a and b are randomly generated numbers distributed uniformly over the range (0,1) using Eq. (4).
Step 4: permute arbitrary changes in position of $oldP$.
Step 5: generate the mutant M according to Eq. (6).
Step 6: generate the trial individuals V according to Eq. (7).
Step 7: selection-II: select the population with better fitness from V and P using Eq. (8).
Step 8: update the best solution.
Step 9: **end while**
Step 10: output the best solution.

Generally speaking, a quadratic programming problem can be described as follows.

$$\begin{aligned} \text{Minimize} \quad & \tfrac{1}{2} d_k^{\mathrm{T}} H_k d_k + \nabla f(x_k)^{\mathrm{T}} d_k, \\ \text{Subject to} \quad & [\nabla g(x_k)]^{\mathrm{T}} d_k + g_i(x_k) = 0, \qquad i = 1, 2, \ldots, m_e; \\ & [\nabla g(x_k)]^{\mathrm{T}} d_k + g_j(x_k) \leqslant 0, \\ & j = m_e + 1, m_e + 2, \ldots, m, \end{aligned} \tag{9}$$

where d_k is the search direction at the kth iteration, $f(x)$ is the objective function subject to the constraints $g(x)$, m_e and m are the number of equality and total constraints respectively, and H_k is the Hessian matrix of Lagrangian function defined by Eq. (10) approximated by B_k according to the quasi-Newton method at the kth iteration shown in Eqs. (11)–(13).

$$L(x_k, \lambda) = f(x_k) + \sum_{j=1}^{m} \lambda_j g_j(x_k). \tag{10}$$

BFGS quasi-Newton method can be represented as follows, where λ is the estimation of the Lagrangian multiplier.

$$B_{k+1} = B_k + \frac{q_k q_k^{\mathrm{T}}}{q_k^{\mathrm{T}} s_k} - \frac{(B_k s_k)(B_k s_k)^{\mathrm{T}}}{s_k^{\mathrm{T}} B_k s_k}. \tag{11}$$

$$q_k = \nabla f(x_{k+1}) + \sum_{i=1}^{m} \lambda_i g_i(x_{k+1}) - \nabla f(x_k) - \sum_{j=1}^{m} \lambda_j g_j(x_k). \quad (12)$$

$$s_k = x_{k+1} - x_k. \quad (13)$$

The merit function is described in the following equation.

$$\psi(x) = L + f(x) + \sum_{i=1}^{m} \lambda_i (g_i(x) - s_i) + \frac{1}{2} \sum_{i=1}^{m} \rho_i (g_i(x) - s_i)^2, \quad (14)$$

where s is the non-negative slack variable, and ρ denotes penalty parameter.

During each iteration process, SQP solves a quadratic programming sub-problem forming as Eq. (9) to form a search direction d_k for a line search procedure, determines the step length according to the merit function described in Eq. (14), and repeats these steps until the solution of the given problem is obtained.

The matlab optimization toolbox is used to minimize the problem using the sequential quadratic programming and the procedure of SQP can be summarized in Algorithm 2.

Algorithm 2 Sequential quadratic programming

Inputs: initial solution x_0, the maximum iteration time $MaxIter$

Step 1: $k = 0, d_k = 1$;

Step 2: **while** $k < MaxIter$ **do**

Step 3: calculate Langrangian function

Step 4: Hessian matrix H_k is approximated by B_k using BFGS quasi-Newton method according to Eqs. (11)–(13).

Step 5: solve quadratic programming sub-problem described in Eq. (9) to obtain search direction d_k.

Step 6: the step length α_k is determined in order to produce a sufficient decrease in the merit function $\psi(x)$ described in Eq.(14).

Step 7: update optimization parameter using $x_{k+1} = x_k + \alpha_k \cdot d_k$.

Step 8: $k = k + 1$;

Step 9: **end while**

Output: x_k

3 SQP enhanced backtracking search algorithm (SQPBSA)

BSA is a global stochastic search algorithm and adaptable to various types of problems, especially when solving multi-model optimization problems. However, BSA suffers low convergence in the later generation stage. SQP belongs to deterministic optimization algorithm, which converts the complex nonlinear optimization problem into a series of quadratic programming sub-problems. The final solution is obtained by solving the quadratic programming sub-problems. Due to the utilization of derivative information, SQP has advantage of fast speed while is easy to stuck into local optima. In the light of the framework of [13–15], and the characteristics of BSA and SQP, we propose SQPBSA. BSA explores the solution as a global search engine. Then, a point generated from BSA is used as an initial input for SQP. Through solving a series of quadratic programming sub-problems, a better solution nearby the initial point is obtained. Therefore, the results obtained from BSA is updated by means of SQP. In SQPBSA, the stop creation is set such that if the certain function evaluations related to the dimension of specific problem is reached. BSA shows strong exploration ability at early evolutionary stage. As a result, SQP is embeded on the early phase of BSA process. Moreover, SQPBSA has to weigh the cost of the SQP function evaluations since SQP is an iterative method. SQP is employed for limited times randomly during evolutionary process.

Algorithm 3 summarizes the steps for combining BSA with SQP. The proposed algorithm SQPBSA is simple and easy to implement. The parameter p divides the evolutionary process into early and later stages. It is at the early stage when current evaluation time is less than the result obtained from p multiplied by the maximum number of iterations $MaxFes$. Otherwise, it is at the later stage. BSA works in the whole process of evolution while the local search technique SQP involves an individual randomly selected from the current population during the early stage in the evolutionary process. Based on the gradient descent feature of SQP, the local search ability of the proposed algorithm is enhanced.

Algorithm 3 Sequential quadratic programming enhanced backtracking search algorithm

Step 1: initialize population size N, stage control parameter p, local search probability $lsrate$, crossover parameter $dimRate$, total number of SQP function evaluation times $innerFes$ and total number of function evaluation times $MaxFes$.

Step 2: initialize population P, and historical population $oldP$ using Eqs. (2) and (3), respectively.

Step 3: evaluate the population P.

Step 4: $Fes = 0$;

Step 5: **while** the stop condition is not satisfied **do**

Step 6: search solution using BSA according to Eqs. (4) and (5).

Step 7: if the evolutionary process is at the early stage and $r < lsrate$ where r is a random number generated from uniformly distributed [0, 1], then randomly select an individual from P updated by SQP.

Step 8: if local search SQP has not been called at the end of the early stage then select an individual from P randomly to be updated by SQP.

Step 9: generate trial population after performing mutation and crossover operators according to Eqs. (6) and (7).

Step 10: evaluate individuals in trial population.

Step 11: update P according to Eq. (8) and select the best individual X_{best}.

Step 12: **end while**

Output: X_{best}.

The proposed method adapts SQP with a certain probability in the early stage. It is necessary to note that SQP is called during the early stage in order to produce a good evolutionary direction guiding the population toward the best solution as soon as possible. Then BSA continues searching better solution based on the local search solution. If it fails to carry out the SQP method in the early stage, SQP is called immediately at the end of the early iterations. SQPBSA focuses on different aspects during two stages. Global and local search is adopted at the early process while only global search BSA works at the later stage. The strategy saves evaluation costs and reduces the running time. Moreover, an individual randomly selected for SQP expands the diversity of the population and prevents the algorithm falling into local optima.

4 Experiments

4.1 Benchmark test functions

In this section, we have carried out experiments of SQPBSA on CEC-2013 [27] test suites including 28 benchmark functions which are tested widely in evolutionary computation research area in order to evaluate its performance. The test functions can be divided into three classes: uni-modal functions $F1 - F5$, basic multimodal functions $F6 - F20$ and composition functions $F21 - F28$. More information of these functions can be found in [27].

4.2 Parameter settings

For each benchmark function, 25 independent runs are performed. The population size N is equal to the dimension D when D is 30 or 50, while it is 30 in the case of $D = 10$. Other parameters are given in Table 1 obtained by trial and error tests. In Table 1, *dimRate* is the crossover rate in BSA, and the stage control parameter p represents the percentage of evaluation times during the early stage accounted of total function evaluations. In addition, *lsRate* is the probability of SQP performed, and *innerFes* is the maximum generation of SQP. The algorithm is stopped if stopping criterion is met. The stopping criterion is a maximum of 100,000, 300,000, 500,000 iterations at dim equal to 10, 30, 50 respectively for each run.

It is necessary to note that parameter *lsrate* is expected to be set to a small value so that SQPBSA invokes SQP with low probability during the early evolutionary stage. The population can easily trap into the local optima if the local search technique SQP is called frequently based on a high value of *lsrate*. There is no significant difference when *lsrate* is set to different small values between 0.01 and 0.05. In this paper, we set *lsrate* to 0.01. We discuss the effectiveness of parameter p and *innerFes* in Section 5.

Table 1 Parameter values

Parameter	*dimRate*	*p*	*lsRate*	*innerFes*
Values	1	0.45	0.01	10,000

We select two indicators in [14] to evaluate the algorithm performance. The error value is defined as $f(X) - f(X^*)$, where X is obtained by the algorithms and X^* is the global optimum of problem. The expression $f(X)-f(X^*)$ means differences between the solution found by the algorithm and the known optimal solution. AVG_{Er} and STD_{Er} represent the average and standard deviation of error values respectively in all 25 runs, and $AVG_{Er} \pm STD_{Er}$ stands for the average and standard deviation of the best error values presented in the following tables. The algorithm is proved to be better if the above indicators are closer to zero. Error values smaller than 10^{-8} are taken as zero [27].

We use Wilcoxon signed-rank test at the 5% significance level to examine whether there is significant difference between two algorithms. The total number of statistical significant cases is given at the bottom of tables.

4.3 Effectiveness of local search technique SQP

Firstly, Table 2 summarizes the results SQPBSA performed on CEC-2013 benchmark test suite. The results obtained by SQPBSA which are relatively accurate are marked in Italics. It can be obtained from the analysis of AVG_{Er} that the proposed algorithm SQPBSA reaches the exact optimal solution for 3 out of 28 functions i.e., $F1$, $F5$, $F6$. SQPBSA is able to find solution near the optimal solution by controlling error less than 10 for seven functions such as $F2$, $F4$, $F10$, $F11$, $F14$, $F16$ and $F19$. For eight problems $F7$, $F8$, $F9$, $F12$, $F17$, $F18$, $F20$, $F22$, the error is below 10^2 and the obtained solution approximates the global optima. For seven complex functions $F13$, $F21$, $F24$, $F25$, $F26$, $F27$, $F28$, the error value obtained by SQPBSA is less than 10^3. The general location of the optimal solution can be learned. With the increasing complexity of the problem, there are multiple local optima. We infer that it is not sufficient enough to produce a good direction guiding the population towards the global optima for the complex multimode problem based on the invocation of SQP at the early stage. For the remaining problems $F3$, $F15$ and $F23$, SQPBSA fails to reach the optimal solution with the error value less than 10^6, 10^4 and 10^4.

Secondly, in order to show the effect of the proposed method, Table 2 also shows the results obtained by BSA when $dim = 30$. The best values obtained by SQPBSA, which are better than the compared algorithms, are marked in boldface.

Table 2 Error values obtained by BSA and SQPBSA for 30-dimensional CEC-2013 benchmark functions

Function	BSA $AVG_{Er} \pm STD_{Er}$	SQPBSA $AVG_{Er} \pm STD_{Er}$	Winner	P-value
F_1	1.01e−30 ± 3.49e−30	*6.69e−30 ± 1.67e−29*	=	0.187500
F_2	1.37e+06 ± 5.35e+05	***7.70e−05 ± 2.35e−05***	+	0.000012
F_3	4.54e+06 ± 4.60e+06	**3.39e+05 ± 7.83e+05**	+	0.000012
F_4	1.27e+04 ± 3.58e+03	***7.12e−05 ± 5.88e−05***	+	0.000012
F_5	0.00e+00 ± 0.00e+00	*0.00e+00 ± 0.00e+00*	=	1.000000
F_6	2.74e+01 ± 2.47e+01	***9.79e−13 ± 3.54e−12***	+	0.000012
F_7	6.82e+01 ± 1.35e+01	**6.75e+01 ± 1.13e+01**	=	0.353258
F_8	2.09e+01 ± 6.72e−02	2.09e+01 ± 5.63e−02	=	0.946369
F_9	2.73e+01 ± 2.75e+00	**2.61e+01 ± 3.02e+00**	=	0.300241
F_{10}	1.90e−01 ± 1.42e−01	***7.95e−03 ± 4.17e−03***	+	0.000012
F_{11}	7.96e−02 ± 2.75e−01	***3.98e−02 ± 1.99e−01***	=	0.500000
F_{12}	8.71e+01 ± 2.14e+01	8.83e+01 ± 1.90e+01	=	0.339479
F_{13}	1.49e+02 ± 2.53e+01	1.52e+02 ± 2.87e+01	=	0.657069
F_{14}	3.56e+00 ± 1.73e+00	*4.67e+00 ± 2.37e+00*	−	0.001303
F_{15}	3.81e+03 ± 4.16e+02	**2.95e+03 ± 4.39e+02**	+	0.000058
F_{16}	1.26e+00 ± 1.66e−01	***1.10e−01 ± 3.73e−02***	+	0.000012
F_{17}	3.09e+01 ± 1.75e−01	3.12e+01 ± 2.35e−01	−	0.001885
F_{18}	1.16e+02 ± 1.99e+01	**8.65e+01 ± 1.48e+01**	+	0.000072
F_{19}	1.07e+00 ± 2.11e−01	*1.21e+00 ± 2.48e−01*	−	0.018555
F_{20}	1.14e+01 ± 4.91e−01	**1.12e+01 ± 5.84e−01**	=	0.967806
F_{21}	2.67e+02 ± 8.00e+01	**2.54e+02 ± 6.36e+01**	=	0.464480
F_{22}	4.33e+01 ± 1.72e+01	**4.19e+01 ± 1.89e+01**	=	0.411840
F_{23}	4.36e+03 ± 5.00e+02	**3.91e+03 ± 3.55e+02**	+	0.003507
F_{24}	2.33e+02 ± 1.03e+01	2.39e+02 ± 1.19e+01	=	0.475825
F_{25}	2.89e+02 ± 8.80e+00	2.90e+02 ± 1.04e+01	=	0.115475
F_{26}	2.00e+02 ± 1.32e−02	**2.00e+02 ± 1.49e−02**	+	0.000065
F_{27}	8.89e+02 ± 1.45e+02	9.12e+02 ± 1.41e+02	=	0.396679
F_{28}	3.00e+02 ± 1.95e−13	3.00e+02 ± 1.69e−13	=	1.000000
+/=/−				10/15/3

For unimodal functions $F1 - F5$, it can be found that SQPBSA reaches the global optimal result for $F1$ and $F5$, and gets high quality solution for $F2$, $F3$ and $F4$. For 15 basic multimodal functions $F6 - F20$, SQPBSA shows better performance for 9 out of 15 functions according to mean error values. However, the remaining 3 out of 6 function results obtained by SQPBSA exhibit inferior performance to BSA based on the Wilcoxon results. For composition functions $F21 - F28$, SQPBSA brings superior solutions for 4 out of 8 functions, and they are not significant at $F21$ and $F22$ according to the Wilcoxon results. According to "+/ = /−", SQPBSA wins and ties BSA on 10 and 15 out of 28 benchmark functions.

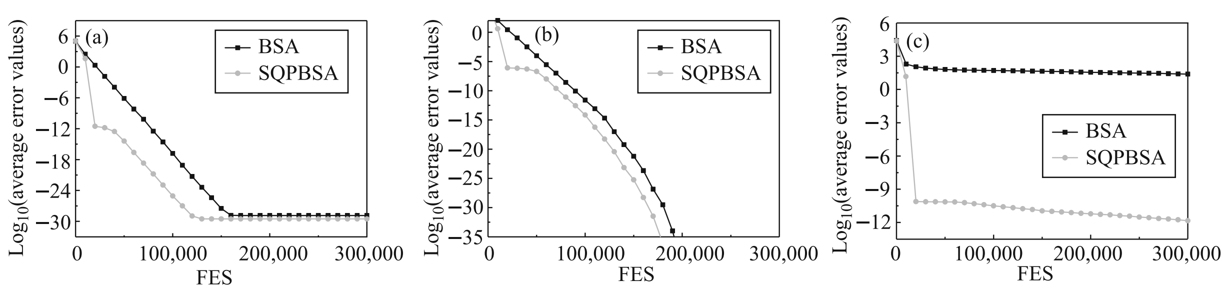
Finally, in order to further investigate the convergence speed of SQPBSA, we plot convergence curves for six selected test function $F1$, $F5$, $F6$, $F20$, $F23$ and $F28$. In the following Fig. 1, the x-coordinate represents the iteration and the y-coordinate represents the mean value of AVG_{Er} taken the logarithm in all 25 runs. We can find that the y-value decreases rapidly which is due to the gradient feature of SQP algorithm. Hence, SQPBSA is able to calculate a promising evolutionary direction to save function evaluations.

4.4 Scalability of SQPBSA

We set the dimension value of benchmark functions at 10 and 50 at each run to analyze the scalability of SQPBSA. The results are listed in Tables 3 and 4. It can be found that BSA and SQPBSA show the similar performance in the case of dimension equal to 10. The results of Wilcoxon rank test show SQPBSA wins and ties BSA in 9 and 19 out of 28 functions, respectively. In the case of dimension equal to 50, SQPBSA gains better solution to most of benchmark functions. Additionally, SQPBSA wins and ties BSA in 14 and 9 out of 28 functions, respectively. We conclude from the above analysis, SQPBSA possesses good stability. It is worthy of pointing out that we will investigate SQPBSA for the high dimensional problems similar as [28].

4.5 Comparison with other algorithms

Firstly, we compare SQPBSA with other effective state-of-art methods which do not have any combination with BSA in [8], namely ABC [29], PSO2011 [30], CMAES [31, 32], CLPSO [33], SADE [34] and JDE [35]. Under the consideration of fairness and convenience, we choose CEC-2005 test



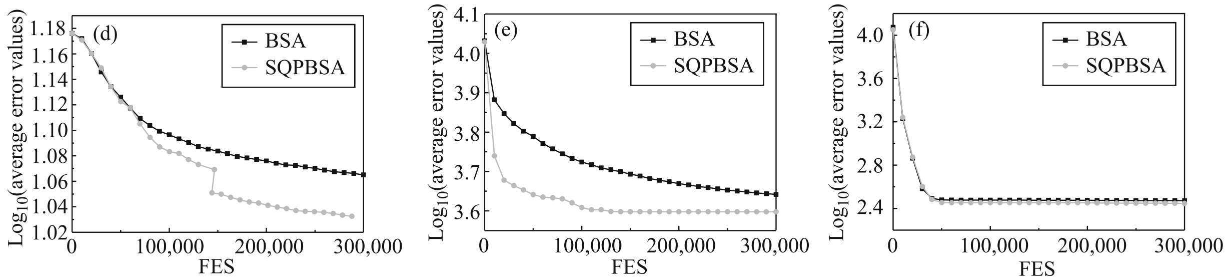


Fig. 1 The convergence curves of BSA and SQPBSA for selected benchmark functions. (a) F_1; (b) F_5; (c) F_6; (d) F_{20}; (e) F_{23}; (f) F_{28}

Table 3 Error values obtained BSA and SQPBSA for 10-dimensional CEC-2013 benchmark functions

Function	BSA $AVG_{Er} \pm STD_{Er}$	SQPBSA $AVG_{Er} \pm STD_{Er}$	Winner	P-value
F_1	0.00e+00±0.00e+00	0.00e+00±0.00e+00	=	1
F_2	2.45e+04±2.58e+04	**7.30e−06±1.93e−05**	+	0.000012
F_3	3.37e+03±7.68e+03	**4.67e+01±2.30e+02**	+	0.000012
F_4	7.84e+02±5.75e+02	**1.91e−04±3.31e−04**	+	0.000012
F_5	0.00e+00±0.00e+00	0.00e+00±0.00e+00	=	1
F_6	4.25e−01±1.96e+00	**3.94e−01±1.96e+00**	+	0.00098
F_7	7.91e+00±7.35e+00	**7.74e+00±6.04e+00**	=	0.882352
F_8	2.03e+01±8.56e−02	**2.02e+01±1.77e−01**	+	0.026431
F_9	3.67e+00±9.36e−01	3.94e+00±1.05e+00	=	0.300241
F_{10}	9.37e−02±3.24e−02	**4.65e−02±2.60e−02**	+	0.000101
F_{11}	0.00e+00±0.00e+00	0.00e+00±0.00e+00	=	1
F_{12}	9.82e+00±2.92e+00	1.17e+01±4.52e+00	=	0.115475
F_{13}	1.57e+01±7.47e+00	**1.46e+01±6.23e+00**	=	0.599802
F_{14}	1.63e−01±6.47e−02	**1.53e−01±7.25e−02**	=	0.840072
F_{15}	6.38e+02±1.34e+02	**5.66e+02±1.21e+02**	=	0.06148
F_{16}	7.34e−01±1.71e−01	**2.55e−01±1.27e−01**	+	0.000012
F_{17}	7.43e+00±2.77e+00	8.30e+00±3.00e+00	=	0.150003
F_{18}	2.45e+01±3.71e+00	**2.03e+01±4.37e+00**	+	0.002064
F_{19}	2.76e−01±8.81e−02	**2.21e−01±1.09e−01**	=	0.065311
F_{20}	2.90e+00±3.40e−01	2.98e+00±2.57e−01	=	0.509755
F_{21}	2.60e+02±1.16e+02	2.12e+02±1.24e+02	=	0.065737
F_{22}	1.50e+01±4.19e+00	**1.30e+01±5.37e+00**	=	0.165837
F_{23}	8.80e+02±1.81e+02	**7.38e+02±1.55e+02**	+	0.008705
F_{24}	1.53e+02±3.41e+01	1.58e+02±3.92e+01	=	0.54491
F_{25}	1.96e+02±2.22e+01	**1.90e+02±2.82e+01**	=	0.26415
F_{26}	1.12e+02±4.84e+00	1.14e+02±6.36e+00	=	0.182896
F_{27}	3.10e+02±2.19e+01	3.21e+02±3.13e+01	=	0.15777
F_{28}	2.29e+02±9.73e+01	**1.81e+02±9.94e+01**	=	0.052512
+/=/−				9/19/0

Table 4 Error values obtained BSA and SQPBSA for 50-dimensional CEC-2013 benchmark functions

Function	BSA $AVG_{Er} \pm STD_{Er}$	SQPBSA $AVG_{Er} \pm STD_{Er}$	Winner	P-value
F_1	2.54e−29±6.38e−29	2.76e−29±5.85e−29	=	0.659912
F_2	2.98e+06±7.39e+05	**8.65e−04±1.89e−04**	+	0.000012
F_3	4.82e+07±3.63e+07	**1.02e+06±2.24e+06**	+	0.000012
F_4	3.17e+04±5.32e+03	**2.81e−05±2.19e−05**	+	0.000012
F_5	5.73e−38±2.48e−37	0.00e+00±0.00e+00	+	0.03125
F_6	4.98e+01±1.43e+01	**3.15e+01±1.96e+01**	+	0.000012
F_7	8.70e+01±7.09e+00	**7.62e+01±6.57e+00**	+	0.000036
F_8	2.11e+01±3.77e−02	2.11e+01±5.17e−02	=	0.353258
F_9	5.43e+01±2.76e+00	5.52e+01±2.59e+00	=	0.26415
F_{10}	4.05e−01±1.42e−01	**4.96e−03±4.54e−03**	+	0.000012
F_{11}	3.98e−02±1.99e−01	7.96e−02±2.75e−01	−	0.000255
F_{12}	2.13e+02±3.65e+01	**1.99e+02±3.23e+01**	=	0.15777
F_{13}	3.21e+02±3.23e+01	3.33e+02±3.80e+01	=	0.220852
F_{14}	2.22e+01±4.44e+00	2.69e+01±4.74e+00	−	0.005816
F_{15}	7.95e+03±8.06e+02	**6.11e+03±4.45e+02**	+	0.00002
F_{16}	1.88e+00±2.54e−01	**9.61e−02±4.72e−02**	+	0.000012
F_{17}	5.43e+01±6.20e−01	5.67e+01±9.97e−01	−	0.000016
F_{18}	2.69e+02±3.75e+01	**1.95e+02±3.96e+01**	+	0.000025
F_{19}	2.60e+00±2.81e−01	3.01e+00±3.41e−01	−	0.000296
F_{20}	2.12e+01±5.04e−01	**2.08e+01±5.51e−01**	+	0.042207
F_{21}	8.05e+02±4.29e+02	**4.61e+02±3.96e+02**	+	0.000482
F_{22}	6.29e+01±1.63e+01	7.87e+01±1.85e+01	−	0.001569
F_{23}	9.64e+03±7.76e+02	**7.77e+03±6.37e+02**	+	0.000012
F_{24}	2.69e+02±1.27e+01	2.73e+02±1.19e+01	=	0.24182
F_{25}	3.81e+02±1.46e+01	3.81e+02±1.61e+01	=	0.946369
F_{26}	2.00e+02±8.62e−02	2.00e+02±1.52e−03	+	0.000012
F_{27}	1.48e+03±2.83e+02	1.50e+03±2.00e+02	=	0.756995
F_{28}	4.00e+02±2.19e−13	4.00e+02±1.43e−13	=	0.225253
+/=/−				14/9/5

suite as benchmark functions and employee the parameter suggested in [36]. More information about these 25 benchmark functions is described in CEC-2005 competition [36].

Table 5 collects the results of PSO2011, CMAES, ABC, JDE, CLPSO, SADE and SQPBSA coping with CEC-2005, and presents the average and the standard deviation of error values. The Wilcoxon test is performed for these seven algorithms on 25 functions and the pairwise comparison results are listed in Table 6. Moreover, the results of Friedman test similarly done in [37] for the compared algorithms are listed in Table 7 to get an overall analysis. From Table 5, it is intuitive to observe that all of the seven algorithms work well according to the average error. PSO2011, CMAES, ABC, JDE, CLPSO, SADE and SQPBSA perform better in 8, 5, 11, 3, 2, 3 and 7 out of 25 functions respectively. For the Wilcoxon test results listed in Table 6, the $R+$ value reflects the degree of SQPBSA superior to the compared algorithm. The symbol "+" means that SQPBSA exhibits better performance signif-

Table 5 Fitness obtained SQPBSA and six non-BSAs for CEC-2005 functions at D=10

Function	Statistics	PSO2011	CMAES	ABC	JDE	CLPSO	SADE	SQPBSA
F_1	Mean	**–450.0000000000000000**	**–450.0000000000000000**	**–450.0000000000000000**	**–450.0000000000000000**	**–450.0000000000000000**	**–450.0000000000000000**	**–450.0000000000000000**
	Std	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
F_2	Mean	**–450.0000000000000000**	**–450.0000000000000000**	–449.9999999999220000	**–450.0000000000000000**	–418.8551838547760000	**–450.0000000000000000**	**–450.0000000000000000**
	Std	0.0000000000000350	0.0000000000000000	0.0000000002052730	0.0000000000000615	51.0880511039985000	0.0000000000000000	0.0000000000000004
F_3	Mean	–44.5873911956554000	**–450.0000000000000000**	387131.2441213970000000	–197.9999999999850000	62142.0000000000000000	245.0483283713550000	–449.9998759279890000
	Std	458.5794120016290000	0.0000000000000000	166951.7336592640000000	391.5169437474990000	34796.1785167236000000	790.6056596723160000	0.0000808008816114
F_4	Mean	**–450.0000000000000000**	77982.4567046980000000	140.4509447125110000	–414.0000000000000000	–178.8320689185280000	**–450.0000000000000000**	–449.9962114268850000
	Std	0.0000000000000460	131376.7365456010000000	217.2646715063190000	55.9309919639279000	394.8667499339530000	0.0000000000000000	0.0056063145455240
F_5	Mean	**–310.0000000000000000**	**–310.0000000000000000**	–291.5327549384120000	–271.0000000000000000	333.4108259915760000	–309.9999999999960000	–309.9999982037400000
	Std	0.0000000000000000	0.0000000000000000	17.6942171217937000	60.5919079609218000	512.6920837704510000	0.0000000000133965	0.0000026247393704
F_6	Mean	393.4959999056240000	390.5315438816460000	391.2531452421960000	**231.3986579112350000**	405.5233436479650000	390.2657719408230000	390.0000000000000000
	Std	16.0224965900462000	1.3783433976373800	3.7254660805238600	247.2968415284400000	10.7480096852869000	1.0114275384776600	0.0000000000000920
F_7	Mean	1091.0644335162500000	1087.2645466786700000	1087.0459486286000000	1141.0459486286000000	1087.0459486286000000	1087.0459486286000000	**1087.045948628590000**
	Std	3.4976948942723200	0.5365230018001780	0.0000000000005585	83.8964879458918000	0.0000000000004264	0.0000000000004814	0.0000000000004618
F_8	Mean	–119.8190232990920000	–119.9261073509850000	–119.7446063439080000	–119.4459380180300000	–119.9300269839980000	–119.7727713703720000	**–119.9999998858070000**
	Std	0.0720107560874199	0.1554021446157740	0.0623866434489108	0.0927418223065644	0.0417913553101429	0.1248514853682450	0.0000000996247817
F_9	Mean	–324.6046006320200000	–306.5782069681560000	**–330.0000000000000000**	–329.8673387923880000	–329.4361898676470000	–329.9668346980970000	**–330.0000000000000000**
	Std	2.5082306041521000	21.9475396048756000	0.0000000000000000	0.3440030182812760	0.6229063711904190	0.1816538397880230	0.0000000000000000
F_{10}	Mean	**–324.3311322538170000**	–314.7871102989330000	–306.7949047862760000	–319.6763749798700000	–321.7278926895280000	–322.9689591871600000	–319.9788220733920000
	Std	3.0072222933667300	8.3115989308305500	5.1787864195870400	4.9173541245304800	1.8971778613701300	2.8254645254663600	3.5602606211767100
F_{11}	Mean	92.5640111212146000	**90.7642785704506000**	94.8428485804138000	93.2972315784963000	94.6109567642977000	91.6859083842723000	94.6637637819396000
	Std	1.5827416781636900	26.4613831425879000	0.6869412813090850	1.8766951726453600	0.6689129174038950	0.9033073777915270	0.7674497674202440
F_{12}	Mean	18611.3142254809000000	–70.0486708747625000	–337.3273080760500000	400.3240208136310000	–447.8870804905020000	–394.5206365378250000	**–459.1997562796040000**
	Std	12508.7866126316000000	637.4585182420270000	56.5730759032367000	688.3344299264300000	11.8934815947019000	128.6353424718180000	2.7697181588050900
F_{13}	Mean	–129.2373581503910000	–128.7850616923410000	–129.8343428775830000	–129.6294851450880000	**–129.8382867796110000**	–129.7129164862680000	–129.6968477405330000
	Std	0.5986210944493790	0.6157633658946230	0.0408016481905455	0.1054759371085400	0.0372256921835666	0.0875456568200232	0.0694143728081848
F_{14}	Mean	**–298.2835926212850000**	–295.1290938304830000	–296.9323391084610000	–296.8839733969750000	–297.5119726691150000	–297.8403738182600000	–296.8347411362020000
	Std	0.5587676271753680	0.1634039984609270	0.2251930667702880	0.4330673614598290	0.3440115280624180	0.4536801689800720	0.2293138462730630
F_{15}	Mean	417.4613663019860000	492.5045364088000000	**120.0000000000000000**	326.6601114362900000	131.3550392249760000	234.2689845349590000	120.0000001125400000
	Std	153.9215808771580000	181.5709657779580000	0.0000000000000188	174.6877238188330000	26.1407360548431000	150.7595974059750000	0.0000005603453462
F_{16}	Mean	**221.4232628350220000**	455.4454684594550000	258.8582688922670000	231.1806131539990000	231.5547154800990000	222.0256674919140000	233.3212271736870000
	Std	12.2450207482898000	254.3583511786970000	11.8823213189685000	13.5473380962764000	11.5441451076421000	6.1841489800660300	10.3120095664262000
F_{17}	Mean	**217.3338617866620000**	681.0349114021570000	265.0370119084380000	228.7309024901770000	240.3635189964930000	221.1801916743850000	242.8664774360980000
	Std	20.6685850658838000	488.0618274343640000	12.4033917090208000	12.3682716268631000	14.8435137485293000	5.7037006844690500	12.4035312596738000
F_{18}	Mean	668.9850326105730000	926.9488078829420000	**513.8925774904480000**	743.9859973770210000	892.4391527217660000	845.4504613493740000	634.4719711596620000
	Std	275.8071370273340000	174.1027182659660000	31.0124861524005000	175.6497294240330000	79.1422224454971000	120.8505129523180000	222.8546759685610000

(*Continued*)

Function	Statistics	PSO2011	CMAES	ABC	JDE	CLPSO	SADE	SQPBSA
F_{19}	Mean	708.2979222913040000	831.2324139697050000	**500.5478931040730000**	776.5150806087790000	863.8929608090610000	809.7183195902260000	573.0953902463740000
	Std	256.2419561521300000	289.7296413284470000	31.2240894705539000	160.7307526692470000	96.5618989087194000	147.3158109824600000	247.5024362248260000
F_{20}	Mean	711.2970397614200000	876.9306161887680000	**483.2984167460740000**	761.2954767038960000	844.6391674419360000	810.5227124472170000	651.1622698435700000
	Std	258.9317052508320000	289.7296413284470000	99.3976740616107000	163.4084080635650000	113.6848457105400000	104.7139423525340000	214.7211010722620000
F_{21}	Mean	1117.8857079625100000	1258.1065536572400000	**659.5351969346130000**	959.3735119754180000	911.4640642691360000	990.8546718748010000	833.6040233658230000
	Std	311.0011859260640000	359.7382897536570000	98.5410511961986000	240.5568407069990000	238.3180009803040000	235.1014092849970000	104.7835010032480000
F_{22}	Mean	1094.8305116977000000	–7.16E+49	**915.4958100611630000**	1133.7536009808600000	1075.5292326436900000	1094.6823697304900000	1009.6443517378100000
	Std	121.3539576317800000	4.39E+50	242.1993331983530000	42.1171260000361000	166.9355145236330000	87.9884000140656000	200.5913287000750000
F_{23}	Mean	1304.3661550124000000	1159.9280867973000000	**830.2290165794410000**	1167.9040488743800000	1070.4327462836400000	1105.2511774948600000	970.4645484659150000
	Std	262.1065863453340000	742.1215416320490000	60.2286903507069000	236.7325108248320000	203.0676627074300000	190.6172874229610000	144.2087373119220000
F_{24}	Mean	500.0000000000000000	653.3355378428050000	**460.0000000000000000**	510.0000000000000000	493.3333333333400000	490.0000000000000000	**460.0000000000000000**
	Std	103.7237710925280000	302.5312999719650000	0.0000000000016493	113.7147065368360000	137.2973951415090000	91.5385729888094000	0.0000000000000000
F_{25}	Mean	1107.9038127876700000	1401.6553278264300000	**930.4565414149210000**	1072.9924659809200000	1258.5157766524700000	1074.3695435628600000	2019.7996395858800000
	Std	127.9566489362040000	253.2428066220210000	87.9959072391079000	2.2606058314671500	241.4024507676890000	2.8314182838917800	6.2472832648032000

icantly than the compared algorithm while symbol "=" indicates that there is no significant difference. Table 6 shows that SQPBSA gets higher $R+$ value 5 out of 6 compared algorithms. At $\alpha = 0.05$, for CMAES and CLPSO, we can obtain that there are significant differences between SQPBSA and these two algorithms. SQPBSA shows better performance. When α is set to 0.1, SQPBSA is significantly superior to PSO2011, CMAES, JDE and CLPSO. To get an overall analysis of seven algorithms, Friedman test is carried out. The average rankings of different algorithms are listed in Table 7. It can be found clearly that SQPBSA is the best, while SADE offers second overall performance, followed by ABC, PSO2011, JDE, CLPSO and CMAES in order. SQPBSA is top ranked since the proposed algorithm achieves good performance in expanded functions $F13$ and $F14$ and hybrid composition functions $F15-F25$. However, for most of these functions, CMAES cannot reap the better performance, resulting in the low ranking.

Table 6 Results of the multiple-problem Wilcoxon test for seven algorithms for CEC-2005 functions at $D = 10$

Algorithm	$R+$	$R-$	P-value	$\alpha = 0.05$	$\alpha = 0.10$
SQPBSA vs. PSO2011	**226.17**	98.83	0.086699	=	+
SQPBSA vs. CMAES	**247.83**	77.17	0.021673	+	+
SQPBSA vs. ABC	147.83	177.17	0.693112	=	=
SQPBSA vs. JDE	**234.83**	90.17	0.051623	=	+
SQPBSA vs. CLPSO	**243.38**	81.63	0.029548	+	+
SQPBSA vs. SADE	**206.83**	118.17	0.232919	=	=

Table 7 Average ranking of seven algorithms by the Friedman test for CEC-2005 functions at $D = 10$

Methods	SQPBSA	SADE	ABC	PSO2011	JDE	CLPSO	CMAES
Ranking	3.12	3.36	3.54	3.96	4.36	4.48	5.18

Secondly, SQPBSA is in comparison with other five algorithms named NBIPOP-aCMA [38], SPSO2011 [39], SPSOABC [40], and PVADE [41] and fk-PSO [42] which were proposed in the CEC-2013 Special Session and Competition on Real-Parameter Single Objective Optimization.

We compare the performance of six algorithms through Friedman test and AVG_{Er} value. The average rankings of the six algorithms calculated by Friedman test are presented in Table 8. As is clearly shown in the table, NBIPOP-aCMA performs best, and SQPBSA offers the second best performance, followed by SPSOABC, fk-PSO, PVADE and SPSO2011. Afterward, Table 9 shows the average and standard deviation of error values. From Table 9, we find that NBIPOP-aCMA, fk-PSO, SPSO2011, SPSOABC, PVADE and SQPBSA perform better in 20, 3, 2, 6, 3 and 8 out of 28 functions respectively. We can see that NBIPOP-aCMA offers the best performance and works well when solving most of the benchmark functions, as it is one of the best top three proposed algorithms during CEC-2013. SQPBSA exhibits the second best performance according to AVG_{Er} value inferior to NBIPOP-aCMA. Overall, the proposed method is capable of solving problems on CEC-2013 and obtaining higher solutions.

Table 8 Average ranking of six algorithms by the Friedman test for CEC-2013 functions at $D = 30$

Methods	NBIPOP-aCMA	SQPBSA	SPSOABC	fk-PSO	PVADE	SPSO2011
Ranking	1.86	2.75	3.39	3.66	4.02	5.32

The above experiments indicates that the proposed method SQPBSA is very efficient in solving benchmark functions in CEC-2005 and CEC-2013.

5 Discussion

5.1 Effect of the parameter p

In the proposed SQPBSA, the stage control parameter p is set to 0.45, which means the function evaluation times in early stage accounts for 45 percent of total evolutionary iteration period. In order to verify the effectiveness of the above choice, we test the algorithm with five different p: 0.15, 0.25, 0.35, 0.45 and 0.55. For each setting, 25 independent runs are performed. Table 10 summarizes the mean error values of the objective function. The Friedman test results are listed in Table 11.

As depicted in Table 10, the mean results provided by $p = 0.45$ are much better than other results for test functions $F6$, $F11$, $F12$, $F15$, $F18$, $F19$ and $F20$, while in the case of $p = 0.45$ the mean errors of test function $F10$, $F24$ and $F25$ are worse than other results. Besides, in the case of $p = 0.45$ the mean results of test functions $F5$, $F8$, $F17$ and $F26$ are similar to those of $p = 0.15, 0.25, 0.35, 0.55$. It is necessary to note that the algorithm with different p equal to 0.15, 0.25, 0.35, 0.45 and 0.55 performs better in 7, 6, 7, 10 and 10 out of 28 functions respectively. In general, the overall performance of $p = 0.45$ is better than that of other settings for p. Table 11 describes the performance of SQPBSA with different p through statistical method Friedman test. It is obvious that $p = 0.45$ offers the best performance, followed by 0.55, 0.35, 0.25 and 0.15. It can be inferred that the local search ability of the proposed method is sufficient when $p = 0.45$ since the fast decline phase is determined by the parameter p and SQP is able to determine a better evolutionary direction.

Table 9 Error values obtained by SQPBSA and five compared algorithms for CEC-2013 benchmark functions at D = 30

Function	NBIPOP-aCMA	fk-PSO	SPSO2011	SPSOABC	PVADE	SQPBSA
	$AVG_{Er} \pm STD_{Er}$	$AVG_{Er} \pm STD_{Er}$	$AVG_{Er} \pm STD_{Er}$	$AVG_{Er} \pm STD_{Er}$	$AVG_{Er} \pm STD_{Er}$	$AVG_{Er} \pm STD_{Er}$
F_1	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**	**6.69e−30±1.67e−29**
F_2	**0.00E+00±0.00E+01**	1.59E+06±8.03E+05	3.38E+05±1.67E+05	8.78E+05±1.69E+06	2.12E+06±1.56E+06	7.70e−05±2.35e−05
F_3	**0.00E+00±0.00E+02**	2.40E+08±3.71E+08	2.88E+08±5.24E+08	5.16E+07±8.00E+07	1.65E+03±2.83E+03	3.39e+05±7.83e+05
F_4	**0.00E+00±0.00E+03**	4.78E+02±1.96E+02	3.86E+04±6.70E+03	6.02E+03±2.30E+03	1.70E+04±2.85E+03	7.12e−05±5.88e−05
F_5	**0.00E+00±0.00E+04**	**0.00E+00±0.00E+00**	5.42E−04±4.91E−05	**0.00E+00±0.00E+00**	1.40E−07±1.86E−07	**0.00E+00±0.00E+00**
F_6	**0.00E+00±0.00E+05**	2.99E+01±1.76E+01	3.79E+01±2.83E+01	1.09E+01±1.09E+01	8.29E+00±5.82E+00	**9.79e−13±3.54e−12**
F_7	2.31E+00±6.05E+00	6.39E+01±3.09E+01	8.79E+01±2.11E+01	5.12E+01±2.04E+01	**1.29E+00±1.22E+00**	6.75e+01±1.13e+01
F_8	**2.09E+01±4.80E−02**	**2.09E+01±6.28E−02**	**2.09E+01±5.89E−02**	**2.09E+01±4.92E−02**	**2.09E+01±4.82E−02**	**2.09e+01±5.63e−02**
F_9	**3.30E+00±1.38E+00**	1.85E+01±2.69E+00	2.88E+01±4.43E+00	2.95E+01±2.62E+00	6.30E+00±3.27E+00	2.61e+01±3.02e+00
F_{10}	**0.00E+00±0.00E+00**	2.29E−01±1.32E−01	3.40E−01±1.48E−01	1.32E−01±6.23E−02	2.16E−02±1.36E−02	7.95e−03±4.17e−03
F_{11}	3.04E+00±1.41E+00	2.36E+01±8.76E+00	1.05E+02±2.74E+01	**0.00E+00±0.00E+00**	5.84E+01±1.11E+01	3.98e−02±1.99e−01
F_{12}	**2.91E+00±1.38E+00**	5.64E+01±1.51E+01	1.04E+02±3.54E+01	6.44E+01±1.48E+01	1.15E+02±1.14E+01	8.83e+01±1.90e+01
F_{13}	**2.78E+00±1.45E+00**	1.23E+02±2.19E+01	1.94E+02±3.86E+01	1.15E+02±2.24E+01	1.31E+02±1.24E+01	1.52e+02±2.87e+01
F_{14}	8.10E+02±3.60E+02	7.04E+02±2.38E+02	3.99E+03±6.19E+02	1.55E+01±6.13E+00	3.20E+03±4.38E+02	**4.67e+00±2.37e+00**
F_{15}	**7.65E+02±2.95E+02**	3.42E+03±5.16E+02	3.81E+03±6.94E+02	3.55E+03±3.04E+02	5.16E+03±3.19E+02	2.95e+03±4.39e+02
F_{16}	4.40E−01±9.26E−01	8.48E−01±2.20E−01	1.31E+00±3.59E−01	1.03E+00±2.01E−01	2.39E+00±2.66E−01	**1.10e−01±3.73e−02**
F_{17}	3.44E+01±1.87E+00	5.26E+01±7.11E+00	1.16E+02±2.02E+01	**3.09E+01±1.23E−01**	1.02E+02±1.17E+01	3.12e+01±2.35e−01
F_{18}	**6.23E+01±4.56E+01**	6.81E+01±9.68E+00	1.21E+02±2.46E+01	9.01E+01±8.95E+00	1.82E+02±1.20E+01	8.65e+01±1.48e+01
F_{19}	2.23E+00±3.41E−01	3.12E+00±9.83E−01	9.51E+00±4.42E+00	1.71E+00±4.68E−01	5.40E+00±8.10E−01	**1.21e+00±2.48e−01**
F_{20}	1.29E+01±5.98E−01	1.20E+01±9.26E−01	1.35E+01±1.11E+00	**1.11E+01±7.60E−01**	1.13E+01±3.28E−01	1.12e+01±5.84e−01
F_{21}	**1.92E+02±2.72E+01**	3.11E+02±7.92E+01	3.09E+02±6.80E+01	3.18E+02±7.53E+01	3.19E+02±6.26E+01	2.54e+02±6.36e+01
F_{22}	8.38E+02±4.60E+02	8.59E+02±3.10E+02	4.30E+03±7.67E+02	8.41E+01±3.90E+01	2.50E+03±3.86E+02	**4.19e+01±1.89e+01**
F_{23}	**6.67E+02±2.90E+02**	3.57E+03±5.90E+02	4.83E+03±8.23E+02	4.18E+03±5.62E+02	5.81E+03±5.04E+02	3.91e+03±3.55e+02
F_{24}	**1.62E+02±3.00E+01**	2.48E+02±8.11E+00	2.67E+02±1.25E+01	2.51E+02±1.43E+01	2.02E+02±1.40E+00	2.39e+02±1.19e+01
F_{25}	**2.20E+02±1.11E+01**	2.49E+02±7.82E+00	2.99E+02±1.05E+01	2.75E+02±9.76E+00	2.30E+02±2.08E+01	2.90e+02±1.04e+01
F_{26}	**1.58E+02±3.00E+01**	2.95E+02±7.06E+01	2.86E+02±8.24E+01	2.60E+02±7.62E+01	2.18E+02±4.01E+01	2.00e+02±1.49e−02
F_{27}	**4.69E+02±7.38E+01**	7.76E+02±7.11E+01	1.00E+03±1.12E+02	9.10E+02±1.62E+02	3.26E+02±1.14E+01	9.12e+02±1.41e+02
F_{28}	**2.69E+02±7.35E+01**	4.01E+02±3.48E+02	4.01E+02±4.76E+02	3.33E+02±2.32E+02	3.00E+02±2.24E−02	3.00e+02±1.69e−13

Based on the above discussion, it is clear that $p = 0.45$ is a reasonable choice for SQPBSA.

5.2 Effect of the parameter $innerFes$

The parameter $innerFes$ in the proposed method is assigned value 10^4, which means the maximum function evaluation time of SQP for each call is up to 10^4. The algorithm is tested with four different $innerFes$ 10^2, 10^3, 10^4 and 10^5 to verify the effectiveness of the parameter adopted in the paper. For each setting, 25 independent runs are performed. The mean error values of the objective function are listed in Table 12. Table 13 describes the Friedman test results.

We can see from Table 12, the mean error values provided by $innerFes = 10^4$ are much better than other results for test functions $F2$, $F6$, $F9$, $F12$, $F15$, $F16$, $F19$, $F20$ and $F21$. However, the mean errors of test function $F24$, $F25$ and $F27$ are worse than other results with $innerFes = 10^4$. In the case of $innerFes = 10^4$, the mean results of test functions $F1$, $F17$ and $F28$ are approximate to those of $innerFes = 10^2$, 10^3 and 10^5. Moreover, it is intuitive to observe that SQPBSA with different $innerFes$ equal to 10^2, 10^3, 10^4 and 10^5 performs better in 11, 9, 14 and 9 out of 28 functions respectively. The results indicate that the overall performance of $innerFes = 10^4$ is better than that of other settings for $innerFes$.

The results of SQPBSA with different $innerFes$ through Friedman test are described in Table 13. As is clearly shown in the table, $innerFes = 10^4$ offers the best performance, followed by 10^3, 10^5 and 10^2. We infer from Table 12 that local search ability is enhanced constantly when $innerFES$ is increased from 10^2 to 10^4. The local exploitation abilities perform best in the case of $innerFes = 10^4$. The algorithm is easy to trap into the local optimal solution when $innerFes$ is increased to 10^5 so that the accuracy of SQPBSA declines.

Based on the above analysis, $innerFes = 10^4$ is a reasonable setting for SQPBSA.

5.3 Running time comparison

Algorithms have been implemented in 64-bit matlabR2011a on a PC (Intel Core i7-4790 CPU, 3.60GHz, 8 GB RAM, 64-

bit Windows 7 operation system). Table 14 reports the running time of SQPBSA similar done in [27, 36]. The calculation of running time is described as follows. First, time of certain test program run on system is obtained as $T0$. The computation time of Function 14 for 200,000 evaluations is $T1$. Then, the average complete computation time of Function 14 with 200,000 evaluations for five times is obtained as $T2$. Finally, $(\hat{T}2-T1)/T0$ is calculated to reflect the complexity of the algorithm. From Table 14, it is clear that additional time is required for optimization when the number of dimensions is increased.

Table 10 Experimental results on 28 benchmark functions with varying stage control parameter

Function	0.15	0.25	0.35	0.45	0.55
F_1	5.68E−30	4.54E−30	1.41E−29	6.69E−30	**4.04E−30**
F_2	1.10E-04	9.94E-05	9.08E-05	7.70E-05	**7.31E−05**
F_3	7.04E+05	7.51E+05	2.39E+05	3.39E+05	**2.04E+05**
F_4	1.76E-04	1.05E-04	**6.61E-05**	7.12E−05	7.29E-05
F_5	2.02E−30	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F_6	8.87E-12	1.06E+00	1.25E-12	**9.79E−13**	2.55E-12
F_7	7.27E+01	6.86E+01	6.59E+01	6.75E+01	**6.15E+01**
F_8	**2.09E+01**	2.10E+01	**2.09E+01**	**2.09E+01**	**2.09E+01**
F_9	2.65E+01	**2.59E+01**	2.67E+01	2.61E+01	2.65E+01
F_{10}	6.71E-03	**5.92E−03**	6.42E-03	7.95E-03	6.41E-03
F_{11}	1.19E-01	1.99E-01	1.19E-01	**3.98E−02**	7.96E-02
F_{12}	9.02E+01	8.96E+01	9.00E+01	**8.83E+01**	9.11E+01
F_{13}	1.55E+02	**1.43E+02**	1.55E+02	1.52E+02	1.52E+02
F_{14}	**3.81E+00**	4.46E+00	4.51E+00	4.67E+00	5.48E+00
F_{15}	3.35E+03	3.38E+03	3.04E+03	**2.95E+03**	3.16E+03
F_{16}	2.41E-01	1.42E-01	1.18E-01	1.10E−01	**7.75E-02**
F_{17}	**3.11E+01**	**3.11E+01**	**3.11E+01**	3.12E+01	3.13E+01
F_{18}	1.07E+02	9.33E+01	9.50E+01	**8.65E+01**	8.68E+01
F_{19}	1.29E+00	1.33E+00	1.28E+00	**1.21E+00**	1.28E+00
F_{20}	1.14E+01	1.14E+01	1.14E+01	**1.12E+01**	1.15E+01
F_{21}	**2.35E+02**	2.37E+02	2.54E+02	2.54E+02	2.58E+02
F_{22}	4.10E+01	4.27E+01	**3.57E+01**	4.19E+01	5.62E+01
F_{23}	4.24E+03	4.09E+03	3.96E+03	3.91E+03	**3.89E+03**
F_{24}	2.34E+02	2.38E+02	2.34E+02	2.39E+02	**2.32E+02**
F_{25}	**2.84E+02**	2.88E+02	2.88E+02	2.90E+02	2.88E+02
F_{26}	**2.00E+02**	**2.00E+02**	**2.00E+02**	**2.00E+02**	**2.00E+02**
F_{27}	**8.32E+02**	8.47E+02	9.43E+02	9.12E+02	8.57E+02
F_{28}	3.00E+02	2.92E+02	**2.84E+02**	3.00E+02	3.00E+02

Table 11 Average ranking on 28 benchmark functions with varying stage control parameter

p	0.15	0.25	0.35	0.45	0.55
Ranking	3.45	3.2	2.88	2.68	2.8

Table 12 Experimental results on 28 benchmark functions with varying stage control parameter

Function	10^2	10^3	10^4	10^5
F_1	**2.02E−30**	**7.57E−30**	**6.69E−30**	**1.01E−29**
F_2	1.43E+06	9.70E+02	**7.70E−05**	7.84E-05
F_3	3.38E+06	7.90E+05	3.39E+05	**1.29E+05**
F_4	4.64E+00	7.27E-05	7.12E-05	**4.67E−05**
F_5	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F_6	1.65E+01	4.35E+00	**9.79E−13**	1.45E-12
F_7	7.30E+01	**6.71E+01**	6.75E+01	7.20E+01
F_8	**2.09E+01**	**2.09E+01**	**2.09E+01**	**2.09E+01**
F_9	2.70E+01	2.74E+01	**2.61E+01**	2.65E+01
F_{10}	7.91E-02	**6.15E−03**	7.95E-03	6.27E-03
F_{11}	**3.98E−02**	7.96E-02	**3.98E−02**	**3.98E−02**
F_{12}	9.12E+01	8.97E+01	**8.83E+01**	9.87E+01
F_{13}	**1.51E+02**	1.56E+02	1.52E+02	1.55E+02
F_{14}	**3.36E+00**	4.22E+00	4.67E+00	5.81E+00
F_{15}	3.40E+03	3.14E+03	**2.95E+03**	3.03E+03
F_{16}	1.28E+00	4.41E-01	**1.10E−01**	1.12E-01
F_{17}	**3.10E+01**	3.11E+01	3.12E+01	3.12E+01
F_{18}	1.06E+02	9.47E+01	8.65E+01	**8.41E+01**
F_{19}	1.23E+00	1.31E+00	**1.21E+00**	1.29E+00
F_{20}	1.17E+01	1.15E+01	**1.12E+01**	1.16E+01
F_{21}	2.71E+02	2.70E+02	**2.54E+02**	2.71E+02
F_{22}	**3.77E+01**	4.17E+01	4.19E+01	4.24E+01
F_{23}	4.11E+03	**3.88E+03**	3.91E+03	3.97E+03
F_{24}	2.33E+02	**2.30E+02**	2.39E+02	2.31E+02
F_{25}	**2.86E+02**	2.88E+02	2.90E+02	**2.86E+02**
F_{26}	**2.00E+02**	**2.00E+02**	**2.00E+02**	**2.00E+02**
F_{27}	**8.54E+02**	8.61E+02	9.12E+02	8.72E+02
F_{28}	3.00E+02	**2.96E+02**	3.00E+02	3.00E+02

Table 13 Average ranking on 28 benchmark functions with varying parameter *innerFes*

innerFES	10^2	10^3	10^4	10^5
Ranking	2.8	2.48	2.14	2.57

Table 14 Running time of SQPBSA on 10, 30 and 50-dimensional CEC-2013 functions

Dimension	$T0$	$T1$	$\hat{T}2$	$(\hat{T}2-T1)/T0$
$D = 10$	0.11	0.26	1.93	15.18
$D = 30$	0.11	0.77	3.81	27.64
$D = 50$	0.11	1.29	5.46	37.91

Table 15 Running time of SQPBSA and five compared algorithms on 10, 30 and 50-dimensional CEC-2013 functions

Dimension	BSA	NBIPOP-aCMA	SQPBSA	SPSOABC	PVADE	SPSO2011
$D = 10$	9.18	62.39	15.18	33.37	7.42	5.17
$D = 30$	10.55	68.11	27.64	165.74	9.97	5.84
$D = 50$	11	103.79	37.91	578.41	16.01	6.15

Table 15 presents the running time of BSA, SQPBSA, NBIPOP-aCMA, SPSOABC, PVADE and SPSO2011 according to $(\hat{T}2 - T1)/T0$. The running time of NBIPOP-aCMA, SPSOABC, PVADE and SPSO2011 is derived from [38–41]. We first compare the running time of SQPBSA and BSA. Due to the calculation of Hessian matrix in SQPBSA, SQPBSA is more time-consuming than BSA. Secondly, we analysis the running time of SQPBSA, NBIPOP-aCMA, SPSOABC, PVADE and SPSO2011. NBIPOP-aCMA shows the best performance on CEC-2013 while it has a second highest computational cost. The third-ranked algorithm SPSOABC is the most time-consuming. SQPBSA is the second best algorithm, but less time-consuming than NBIPOP-aCMA and SPSOABC. PVADE and SPSO2011 do not offer the results as competitive as the other algorithms while these two methods save the most computation time.

6 Conclusion

In this paper, BSA combined with SQP for solving numerical optimization problems is proposed called SQPBSA. SQPBSA adopts BSA as a global search engine and SQP algorithm as a local search technique. At each iteration process, the proposed method optimizes one individual randomly selected from population.

Based on the characteristics of BSA powerful global exploration capability and SQP fast gradient decreasing speed, SQPBSA preserves the advantages and makes up for the disadvantages of BSA and SQP.

SQPBSA is verified by effect and stability experiments. The results reveal that sequential quadratic programming enhanced backtracking search algorithm provides competitive and effective results. Moreover, SQPBSA is compared with state-of-the-art evolutionary algorithms solving test functions collected in CEC-2005 and CEC-2013. Meanwhile, the effect of the parameter p and $innerFes$ on the performance of SQPBSA is also investigated. SQPBSA is capable of finding solution within reasonable period of time. The results present that our method is able to solve benchmark test functions and provide promising performance.

In the future, it is interesting to investigate the proposed algorithm for high dimensional problems. We also plan to modify SQPBSA to solve combinational optimization problems.

Acknowledgements This work was supported by the NSFC-Guangdong Joint Fund (U1201258), the National Natural Science Foundation of China (Grant No. 61573219), the Shandong Natural Science Funds for Distinguished Young Scholars (JQ201316), the Fundamental Research Funds of Shandong University (2014JC028), and the Natural Science Foundation of Fujian Province of China (2016J01280).

References

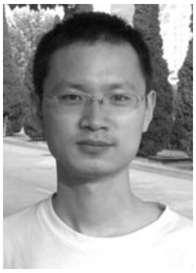
1. Holland J H. Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. Cambridge, Massachusettes: The MIT press, 1992
2. Storn R, Price K. Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012. Berkeley, CA: International Computer Science Institue, 1995
3. Dorigo M, Maniezzo V, Colorni A. Ant system: optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 1996, 26(1): 29–41
4. Kennedy J, Eberhart R C. Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks. 1995, 1942–1948
5. Eberhart R C, Kennedy J. A new optimizer using particle swarm theory. In: Proceedings of the 6th International Symposium on Micro Machine and Human Science. 1995, 39–43
6. Chen W N, Zhang J, Lin Y, Chen N, Zhan Z H, Chung H S H, Li Y, Shi Y H. Particle swarm optimization with an aging leader and challengers. IEEE Transactions on Evolutionary Computation, 2013, 17(2): 241–258
7. Yu W J, Shen M, Chen W N, Zhan Z H, Gong Y J, Lin Y, Liu O, Zhang J. Differential evolution with two-level parameter adaptation. IEEE Transactions on Cybernetics, 2014, 44(7): 1080–1099
8. Civicioglu P. Backtracking search optimization algorithm for numerical optimization problems. Applied Mathematics and Computation, 2013, 219(15): 8121–8144
9. Agarwal S K, Shah S, Kumar R. Classification of mental tasks from eeg data using backtracking search optimization based neural classifier. Neurocomputing, 2015, 166: 397–403
10. Yang D D, Ma H G, Xu D H, Zhang B H. Fault measurement for siso system using the chaotic excitation. Journal of the Franklin Institute, 2015, 352(8): 3267–3284
11. Zhang C J, Lin Q, Gao L, Li X Y. Backtracking search algorithm with three constraint handling methods for constrained optimization problems. Expert Systems with Applications, 2015, 42(21): 7831–7845
12. Zhao W T, Wang L J, Yin Y L, Wang B Q, Wei Y, Yin Y S. An improved backtracking search algorithm for constrained optimization problems. In: Proceedings of the 7th International Conference on Knowledge Science, Engineering and Management. 2014, 222–233
13. Mallick S, Kar R, Mandal D, Ghoshal S. CMOS analogue amplifier circuits optimisation using hybrid backtracking search algorithm with differential evolution. Journal of Experimental & Theoretical Artificial Intelligence, 2016, 28(4): 719–749
14. Wang L T, Zhong Y W, Yin Y L, Zhao W T, Wang B Q, Xu Y L. A hybrid backtracking search optimization algorithm with differential evolution. Mathematical Problems in Engineering, 2015
15. Ali A F. A memetic backtracking search optimization algorithm for economic dispatch problem. Egyptian Computer Science Journal, 2015, 39(2)
16. Qian C, Yu Y, Zhou Z H. Pareto ensemble pruning. In: Proceedings of the 29th AAAI Conference on Artificial Intelligence. 2015, 2935–2941
17. Attaviriyanupap P, Kita H, Tanaka E, Hasegawa J. A hybrid EP and SQP for dynamic economic dispatch with nonsmooth fuel cost function. IEEE Transactions on Power Systems, 2002, 17(2): 411–416
18. Cai J J, Li Q, Li L X, Peng H P, Yang Y X. A hybrid CPSO–SQP method for economic dispatch considering the valve-point effects. Energy Conversion and Management, 2012, 53(1): 175–181
19. Basu M. Hybridization of bee colony optimization and sequential quadratic programming for dynamic economic dispatch. International Journal of Electrical Power & Energy Systems, 2013, 44(1): 591–596

20. Morshed M J, Asgharpour A. Hybrid imperialist competitive-sequential quadratic programming (HIC-SQP) algorithm for solving economic load dispatch with incorporating stochastic wind power: a comparative study on heuristic optimization techniques. Energy Conversion and Management, 2014, 84: 30–40
21. Zhan Z H, Zhang J, Li Y, Shi Y H. Orthogonal learning particle swarm optimization. IEEE Transactions on Evolutionary Computation, 2011, 15(6): 832–847
22. Blum C, Puchinger J, Raidl G R, Roli A. Hybrid metaheuristics in combinatorial optimization: a survey. Applied Soft Computing, 2011, 11(6): 4135–4151
23. Lozano M, García-Martínez C. Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: Overview and progress report. Computers & Operations Research, 2010, 37(3): 481–497
24. Zhang J, Zhan Z H, Lin Y, Chen N, Gong Y J, Zhong J H, Chung H, Li Y, Shi Y H. Evolutionary computation meets machine learning: a survey. Computational Intelligence Magazine, IEEE, 2011, 6(4): 68–75
25. Nocedal J, Wright S. Sequential quadratic programming. In: Sun W Y, Yuan Y X, eds. Optimization Theory and Methods. Springer Optimization and Its Application, Vol 1. Springer Science & Business Media, 2006, 529–533
26. Wilson R B. A simplicial algorithm for concave programming. Dissertation for the Doctoral Degree. Cambridge, MA: Harvard University, 1963
27. Liang J J, Qu B Y, Suganthan P N, Hernández-Díaz A G. Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization. Technical Report. 2013
28. Qian H, Hu Y Q, Yu Y. Derivative-free optimization of high-dimensional non-convex functions by sequential random embeddings. In: Preceedings of the 25th International Joint Conference on Artificial Intelligence. 2016, 1946–1952
29. Karaboga D. An idea based on honey bee swarm for numerical optimization. Technical Report. 2005
30. Clerk M. Standard particle swarm optimisation. Technical Report. 2012
31. Hansen N, Ostermeier A. Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation, 2001, 9(2): 159–195
32. Igel C, Hansen N, Roth S. Covariance matrix adaptation for multi-objective optimization. Evolutionary Computation, 2007, 15(1): 1–28
33. Liang J J, Qin A K, Suganthan P N, Baskar S. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Transactions on Evolutionary Computation, 2006, 10(3): 281–295
34. Qin A K, Suganthan P N. Self-adaptive differential evolution algorithm for numerical optimization. In: Proceedings of IEEE Congress on Evolutionary Computation. 2005, 1785–1791
35. Brest J, Greiner S, Bošković B, Mernik M, Zumer V. Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. IEEE Transactions on Evolutionary Computation, 2006, 10(6): 646–657
36. Suganthan P N, Hansen N, Liang J J, Deb K, Chen Y P, Auger A, Tiwari S. Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. KanGAL Report. 2005
37. Gong W Y, Cai Z H. Differential evolution with ranking-based mutation operators. IEEE Transactions on Cybernetics, 2013, 43(6): 2066–2081
38. Loshchilov I. CMA-ES with restarts for solving CEC 2013 benchmark problems. In: Proceedings of IEEE Congress on Evolutionary Computation. 2013, 369–376
39. Zambrano-Bigiarini M, Clerc M, Rojas R. Standard particle swarm optimisation 2011 at CEC-2013: a baseline for future PSO improvements. In: Proceedings of IEEE Congress on Evolutionary Computation. 2013, 2337–2344
40. El-Abd M. Testing a particle swarm optimization and artificial bee colony hybrid algorithm on the CEC13 benchmarks. In: Proceedings of IEEE Congress on Evolutionary Computation. 2013, 2215–2220
41. Dos Santos Coelho L, Ayala H V H. Population's variance-based adaptive differential evolution for real parameter optimization. In: Proceedings of IEEE Congress on Evolutionary Computation. 2013, 1672–1677
42. Nepomuceno F V, Engelbrecht A P. A self-adaptive heterogeneous PSO for real-parameter optimization. In: Proceedings of IEEE Congress on Evolutionary Computation. 2013, 361–368

Wenting Zhao received her BS degree in electrical engineering from Shandong University (SDU), China in 2014. Currently, she is studying at SDU for a master degree in software engineering. Her main research interests are evolutionary computation and machine learning.

Lijin Wang received his BS and MS degrees from Fujian Agriculture and Forestry University (FAFU), China in 2000 and 2005 respectively, and his PhD degree from Beijing Forestry University, China in 2008. He is currently a post-doctoral fellow with the School of Computer Science and Technology, Shandong University, China. He is also an associate professor with the College of Computer and Information Science, FAFU. His research interests include evolutionary algorithms and intelligent information processing.

Yilong Yin received his PhD degree from Jilin University, China in 2000. From 2000 to 2002, he worked as a postdoctoral fellow in the Department of Electronics Science and Engineering, Nanjing University, China. He is currently the director of MLA Group and a professor of the School of Computer Science and Technology, Shandong University, China. His research interests include machine learning, data mining, and computational medicine.



Bingqing Wang received his BS degree in electrical engineering from Qingdao University, China in 2012. From 2012 to 2016, he received his master degree in School of Computer Science and Technology, Shandong University, China. His main research interests are machine learning and application.

Yuchun Tang received his MD degree majored in sectional and imaging anatomy at Shandong University, China in 2009. He is currently a teacher in Shandong University School of Medicine, China. His research interests include sectional and imaging anatomy, brain imaging, and computational neuroscience.