International Journal of Computational Intelligence Systems
Vol. **12(2)**, 2019, *pp.* 1270–1281
DOI: https://doi.org/10.2991/ijcis.d.191101.004; ISSN: 1875-6891; eISSN: 1875-6883
https://www.atlantis-press.com/journals/ijcis/

A Hybrid Global Optimization Algorithm Based on Particle Swarm Optimization and Gaussian Process

Yan Zhang[1,2], Hongyu Li[1,3,*], Enhe Bao[1], Lu Zhang[1,4], Aiping Yu[1]

[1] *College of Civil Engineering and Architecture, Guilin University of Technology, Guilin 541004, China*
[2] *Guangxi Key Laboratory of Geomechanics and Geotechnical Engineering, Guilin 541004, China*
[3] *Collaborative Innovation Center for Exploration of Hidden Nonferrous Metal Deposits and Development of New Materials in Guangxi, Guilin University of Technology, Guilin 541004, China*
[4] *Department of Civil and Materials Engineering, University of Illinois at Chicago, Chicago, IL 60607, USA*

ARTICLE INFO

Article History
Received 22 Apr 2019
Accepted 28 Oct 2019

Keywords
Swarm optimization
Gaussian process
Global optimization
Surrogate approach

ABSTRACT

The optimization problems and algorithms are the basics subfield in artificial intelligence, which is booming in the almost any industrial field. However, the computational cost is always the issue which hinders its applicability. This paper proposes a novel hybrid optimization algorithm for solving expensive optimizing problems, which is based on particle swarm optimization (PSO) combined with Gaussian process (GP). In this algorithm, the GP is used as an inexpensive fitness function surrogate and a powerful tool to predict the global optimum solution for accelerating the local search of PSO. In order to improve the predictive capacity of GP, the training datasets are dynamically updated through sorting and replacing the worst fitness function solution with the better solution during the iterative process. A numerical study is carried out using twelve different benchmark functions with 10, 20 and 30 dimensions, respectively. Regarding solving of the ill-conditioned computationally expensive optimization problems, results show that the proposed algorithm is much more efficient and suitable than the standard PSO alone.

1. INTRODUCTION

Swarm intelligence [1–4] was identified as a promising new optimization technique over the past decade. It is an attempt to develop algorithms inspired by the collective behavior of social insects and other animal societies. There are many swarm intelligence approaches designed for algorithmic optimization, such as cuckoo search algorithm [5–6], bat algorithm [7–8], artificial bee colony algorithm [9], fish school algorithm [10], ant colony optimization algorithms [11] and particle swarm optimization (PSO) algorithms [12–14]. Though most of swarm intelligence share the similar concept on collective behavior of decentralized, self-organized systems, natural or artificial, each swarm intelligence-based algorithm has its characteristic. Therefore, the different swarm intelligence algorithm could be assigned in different application to maximize the performance. For example, Cuckoo search idealized such breeding behavior, and thus can be applied for series system in hardware design [15]. Bat algorithm mimics a frequency-tuning method to control the dynamic behavior thus can be used to design the transport network [16]. Herein, the PSO as a more general approach developed by Eberhart and Kennedy in 1995 has better applicability over the others. The PSO is a metaheuristic as it requires few or no assumptions about the problems; therefore, it can search very large space of candidate solutions [17]. In the search space, the position and velocity of particle motion will be equated by simple mathematical expression. Specifically, the form of path is relinked among optimal position (*pbest*). In this sense, PSO shares many similarities with evolutionary computation techniques such as genetic algorithms (GAs) [18]. Compared to GA, PSO has fewer parameters to adjust and better convergence capability for the complex optimization problems [19]. However, like other population-based algorithms, PSO has some inevitable disadvantages, for example, it is difficult in defining initial parameters; this method is not applicable for the problems of scattering and it can be trapped into a local minimum especially with complex problems. More important, PSO requires a large number of fitness function evaluations. Moreover, aiming to practical engineering problems, the real function is usually missing. Therefore, the numerical simulation is introduced to build the explicit relationship between variable and fitness [20,21]. But it is often infeasible to apply the process to fitness evaluation due to its high computational cost. Examples of such problems include large-scale finite element method analysis or computational fluid dynamics simulations [22]. In such problems, a single exact fitness function evaluation (involving the analysis of a complex engineering system based on high fidelity simulation codes) often consumes massive CPU time. Therefore, the high computational cost involved posing a serious impediment to PSO's successful application.

In order to reduce the computational cost of PSO, many methods have been considered and implemented. GA can be used to

* *Corresponding author. E-mail:* lihongyu@glut.edu.cn

modify the decision vector to speed up the procedure [23,24]. By applying the binary variables, the PSO can be adapted to consume less computational cost [25]. However, more promising way to significantly reduce the computational cost of PSO is to employ computationally cheap surrogate model or metamodel in place of computationally expensive exact fitness evaluations. The metamodel as a surrogate model is constructed by generating a set of data points in the design space and evaluating the cost function. This method relies on establishing the accurate metamodel. For example, Praveen and Duvigneau [26] developed a low cost PSO by using metamodels to simulate the aerodynamic shape. Selleri *et al.* [27] try to optimize the array in the design of antennas and microwave by introducing metamodel to PSO. Besides metamodels, research on surrogate-assisted evolutionary computation began over a decade ago and had received considerably more interest in recent years [28–32]. In the surrogate approach, a surrogate model is trained on existing evaluated individuals (fitness cases) in order to guide the search for promising solutions. Biswas *et al.* [33] coupled the Bacterial Foraging Optimization Algorithm (BFOA) with PSO for optimizing multimodal and high-dimensional functions. Zhang *et al.* [34] employed the artificial immune system (AIS) and chaos operator as a surrogate to lower the computational cost in PSO. By leveraging surrogate models, the number of expensive fitness evaluations decreases, which result in significant decrease in computational cost [35–37]. Though currently a variety of empirical models can be used to construct the approximation (e.g., polynomial models (PMs), artificial neural networks (ANNs) and radial basis function networks (RBFNs)), there are still limitations which are difficult to be overcome (i) for the surrogate models, PM can't cope with large dimensional multimodal problems since they generally carry out approximation using simple quadratic models; ANN always has difficulty in finding an appropriate network topology and the optimum hyperparameter; a major difficulty in RBFN is to set the optimum decay parameter. Therefore, there is an urgent need to develop a novel framework to solve those problems. (ii) In respect to the strategies of algorithms, the performance of the algorithms is highly dependent on the quality of training data. Generally speaking, to achieve a valid surrogate model, the data for training must be "representative" of the overall design space and the global optimum solution should be included in the interval of training data. Nevertheless, the training dataset for training surrogate models remain unchanged during the whole searching process. To this end, the conventional strategy does not guarantee that the selected training dataset covers the entire design space or the global optimum solution locates in the interval of training data, especially for the scenarios with large variables but with limited training datasets. So, the surrogate models are very easy to be trapped by the local optimum solution if the training datasets are not selected properly.

Gaussian process (GP) can directly capture the model uncertainty; and also, it is be able to add prior knowledge and specifications about the shape of the model by selecting different kernel functions. Therefore, it is suitable to introduce the Gaussian mutation to PSO to improve the efficiency. Higashi and Iba [38] demonstrate a way to combine PSO with GP. The proposed strategy achieves better performance than PSO itself. Krohling [39] developed a Gaussian PSO algorithm to solve the multimodal optimization problems. But the PSO still can be stuck in local minima when optimizing functions. The key to adapt GP in PSO is to find the proper optimal combination strategy. In this process, for most of current methods, it excessively relies on the sample if the approximate model is directly replaced with the fitness function. In particular, the accuracy of regression is determined by the sample. If the representation of learning sample is either poor or off from the global trend, the corresponding accuracy of prediction will be low, leading failed in search optimal value.

In order to overcome the problems described above, a novel hybrid GP surrogate-assisted PSO optimization algorithm is proposed in this paper. Comparing with the conventional surrogate models, the proposed algorithm has merits in the following aspects: (i) training datasets are generated randomly; (ii) the local search is accelerated; (iii) the training datasets can be dynamically updated. To sum up, the proposed algorithm can guarantee the accuracy and efficiency for solving computationally expensive global optimization problems. For convenience, the proposed algorithm will be named as GP-PSO. The remainder of this paper is structured as follows: Section 2 gives a brief introduction of the related methodology including PSO and GP; Section 3 presents the proposed GP-PSO algorithm; Section 4 demonstrates the validity and efficiency of the proposed algorithm by experiments. Finally, the conclusions and future works are presented in Section 5.

2. REVIEW OF RELATED METHODOLOGY

2.1. Particle Swam Optimization

PSO is initialized with a population of random solutions and searches for optima by updating generations. In PSO, the potential solutions, called particles, fly through the problem space by following the current "optimum" particles. There are two "optimum" particles: one keeps track of its coordinates associated with the best solution in current stage. The best solution is defined as *pbest*. The other "best" tracked by the particle swarm optimizer is obtained so far by any particle in the neighbors of the particle. When a particle takes all the population as its topological neighbors, this global best denoted as *gbest*. With the help of the two best values, the particle updates its velocity and positions with following equation:

$$\left.\begin{aligned} v_{id} &= wv_{id} + c_1 r_1 (P_{id} - x_{id}) + c_2 r_2 \left(p_{gd} - x_{id}\right) \\ x_{id} &= x_{id} + v_{id} \end{aligned}\right\}, \tag{1}$$

where v_{id} is a velocity vector, x_{id} is a position vector. p_{id} represents the best ever position of particle i, its fitness is *pbest*. p_{gd} corresponds to the global best position in the process of particle iteration on the *d-th* dimension, its fitness is *gbest*. The parameters r_1 and r_2 are two random values, uniformly distributed in [0, 1] [40]. c_1 and c_2 are acceleration constants, usually $c_1 = c_2 = [1.8, 2.0]$ [41]. The parameter w is the inertia weight, which controls the influence of previous velocity on the new velocity. Inertia weight w can be determined by equation:

$$w = w_{\max} - \frac{w_{\max} - w_{\min}}{t_{\max}} t, \tag{2}$$

where t is the current iteration step, $t_{\max}$ is the maximum iteration step, $w_{\max}$ is the maximum inertia weight, $w_{\min}$ is the minimum inertia weight. Usually, $w_{\max} = 0.9$, $w_{\min} = 0.4$ [42].

2.2. Gaussian Process

For the approximation of the fitness function we chose GP, which is a newly developed machine learning technology based on strict theoretical fundamentals and Bayesian theory [43]. In recent years, GP has attracted much attention in the machine learning community [44–46]. Like ANNs, which are the most prominent surrogate models, GP can approximate any function. During searching the optimum solution by PSO, a GP model is used to approximate the real function, so that the number of evaluations of real functions can be dramatically reduced. Compared to ANN, GP's main advantage is the simplicity: no need to choose network size or topology. Besides, GP can automatically choose the optimum hyper-parameters. More details about GP can be seen in the work by Rasmussen and Williams [47].

A GP is a collection of random variables, any finite set of which have a joint Gaussian distribution. A GP is completely specified by its mean function $m(\boldsymbol{x})$ and the covariance function $k\left(\boldsymbol{x}, \boldsymbol{x}'\right)$ as

$$f(\boldsymbol{x}) \sim GP\left(m\left(\boldsymbol{x}\right), k\left(\boldsymbol{x}, \boldsymbol{x}'\right)\right). \tag{3}$$

There is a training set $\boldsymbol{D}$ of m observations, $\boldsymbol{D} = \left\{\left(\boldsymbol{x}_i, y_i\right) | i = 1, \ldots m\right\}$, where $\boldsymbol{x}$ denotes an input vector with n dimension, y denotes a scalar output or target. The goal of the Bayesian forecasting is to compute the distribution $p\left(y_*|\boldsymbol{x}_*, \boldsymbol{D}\right)$ of output $\boldsymbol{y}_*$ given a test input $\boldsymbol{x}_*$ and a set of training points $\boldsymbol{D}$ [48]. Using the Bayesian rule, the posterior distribution for the GP outputs y_* can be obtained. By conditioning on the observed targets in the training set, the predictive distribution is Gaussian:

$$y_*|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y} \sim N\left(\overline{y}\left(\boldsymbol{x}_*\right), \overline{\sigma}\left(\boldsymbol{x}_*\right)\right), \tag{4}$$

where the mean and variance are given by

$$\overline{y}\left(\boldsymbol{x}_*\right) = \boldsymbol{k}_*^T \left(\boldsymbol{K} + \boldsymbol{\sigma}_\mathbf{n}^\mathbf{2}\boldsymbol{I}\right)^{-1} \boldsymbol{y}, \tag{5}$$

$$\overline{\sigma}^2\left(\boldsymbol{x}_*\right) = k\left(\boldsymbol{x}_*, \boldsymbol{x}_*\right) - \boldsymbol{k}_*^T \left(\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}\right)^{-1} \boldsymbol{k}_*, \tag{6}$$

where a compact form of the notation setting for matrix of the covariance functions are $\boldsymbol{k}_* = \boldsymbol{K}\left(\boldsymbol{X}, \boldsymbol{x}_*\right)$, $\boldsymbol{K} = \boldsymbol{K}\left(\boldsymbol{X}, \boldsymbol{X}\right)$, σ_n^2 is the unknown variance of the Gaussian noise.

GP procedure can handle interesting models by simply using a covariance function with an exponential term:

$$\boldsymbol{k}\left(\boldsymbol{x}_p, \boldsymbol{x}_q\right) = \sigma_f^2 \exp\left(-\frac{\left\|\boldsymbol{x}_p - \boldsymbol{x}_q\right\|^2}{2\boldsymbol{l}^2}\right) + \sigma_n^2 \delta_{pq}, \tag{7}$$

where $\boldsymbol{l}$ is the length-scale vector, σ_f^2 is the signal variance, δ_{pq} is a Kronecker delta. This function expresses the idea that nearby inputs have highly correlated outputs. The GP employs a set of hyper-parameters $\boldsymbol{\theta}$ including the length-scale $\boldsymbol{l}$, the signal variance σ_f^2, the noise variance σ_n^2. The hyper-parameters $\boldsymbol{\theta}$ can be optimized based on log-likelihood framework:

$$L = \log p\left(\boldsymbol{y} | \boldsymbol{X}, \boldsymbol{\theta}\right) = -\frac{1}{2}\boldsymbol{y}^T \boldsymbol{C}^{-1} y - \frac{1}{2}\log \det \boldsymbol{C} - \frac{n}{2}\log 2\pi. \tag{8}$$

The log-likelihood and its derivative with respect to $\boldsymbol{\theta}$ can be expressed as

$$\frac{\partial L}{\partial \boldsymbol{\theta}} = -\frac{1}{2} tr\left(\boldsymbol{C}^{-1}\frac{\partial \boldsymbol{C}}{\partial \boldsymbol{\theta}}\right) + \frac{1}{2} y^T \boldsymbol{C}^{-1}\frac{\partial \boldsymbol{C}}{\partial \boldsymbol{\theta}}\boldsymbol{C}^{-1} y, \tag{9}$$

where $\boldsymbol{C} = \boldsymbol{K} + \boldsymbol{\sigma}_\boldsymbol{n}^\mathbf{2}\boldsymbol{I}$.

Hyper-parameters $\boldsymbol{\theta}$ are initialized to random values in a reasonable range, and then the algorithm searches for the optimal values by using an iterative method such as conjugate gradient.

3. THE PROPOSED GP-PSO ALGORITHM

In this paper, in order to improve the PSO's efficiency for computationally expensive optimization problem, a GP-based PSO algorithm is developed. In GP-PSO algorithm, GP is applied to approximate the real function. Once the approximation function is found, we can directly use the approximation function instead of real function to evaluate the fitness of the particles. Hence, the number of real function evaluations can be reduced greatly by using PSO to solve the optimization problem. Key points of GP-PSO algorithm are the following:

1. Accelerate searching based on approximating real function by GP
   In order to accelerate the local search of PSO, once the *pbest* particle and *gbest* particle in each iterative step are found, some new particles are generated using Eq. (1) and their fitness are estimated by trained GP. Then the best one with the minimum fitness is selected. To eliminate the predictive error of fitness evaluated by GP approximation, the fitness of the best particle is evaluated again using the real function. If the real fitness of the best one is less than *gbest*, it becomes the global best in current iterative step and *gbest* is replaced by its fitness. Thus, the number of real function evaluations in exploration process is only one because the fitness of the new particles with the exception of the best one is evaluated by trained GP rather than real function. So the computational cost of the process of accelerating search is low.
2. Dynamically update the training datasets to improve approximation quality of GP model
   The accuracy of GP-PSO algorithm depends on the appropriateness of the GP model generated, while in aspect of prediction, the accuracy of the GP model in interpolation and validity highly depend on the quality of training datasets. To avoid excessively relying on the initial training datasets and to improve the quality of training datasets gradually in the exploration process, the training datasets for training GP is updated dynamically in GP-PSO algorithm. The strategies can be realized through the following two ways:
   - In each iterative step of PSO, particles of all generation are ranked from small fitness to large fitness, the top $2\times N$ particles and their real fitness are selected as training datasets by multiple try-and-error, where N is the population size. In this case, the Rastrigin function was selected as benchmark due to its high complexity. Through comparison of different dimension of particles (from N to

$3\times N$), $2\times N$ was used because it resulted in lowest computational cost on condition of the required accuracy.

- In each iterative step of PSO, the best particle and its fitness is found by local search, which is based on evaluation of the function generated by the GP model, the worst particle in training datasets is replaced with the best one.

Thus the training datasets are always consistent with the elite group of the particles, the quality of general approximation of GP model can be enhanced in the optimization process. Specifically speaking, the implementation procedure is presented in this section associated with the flow chart, see in Figure 1. Its framework mainly consists of two cycles. The cycle depending on "p" is defined as outer loop and the cycle depending on "k" is defined as inner loop. Meanwhile, the initial values for both p and k are zero.

Specific implementation steps of GP-PSO are shown as follows:

1. Generate N particles in the 1st generation, in which the particles are randomly distributed throughout the design space, which is bounded by specified limits. N is the population size of PSO, it is usually determined by the dimension of optimization problem.

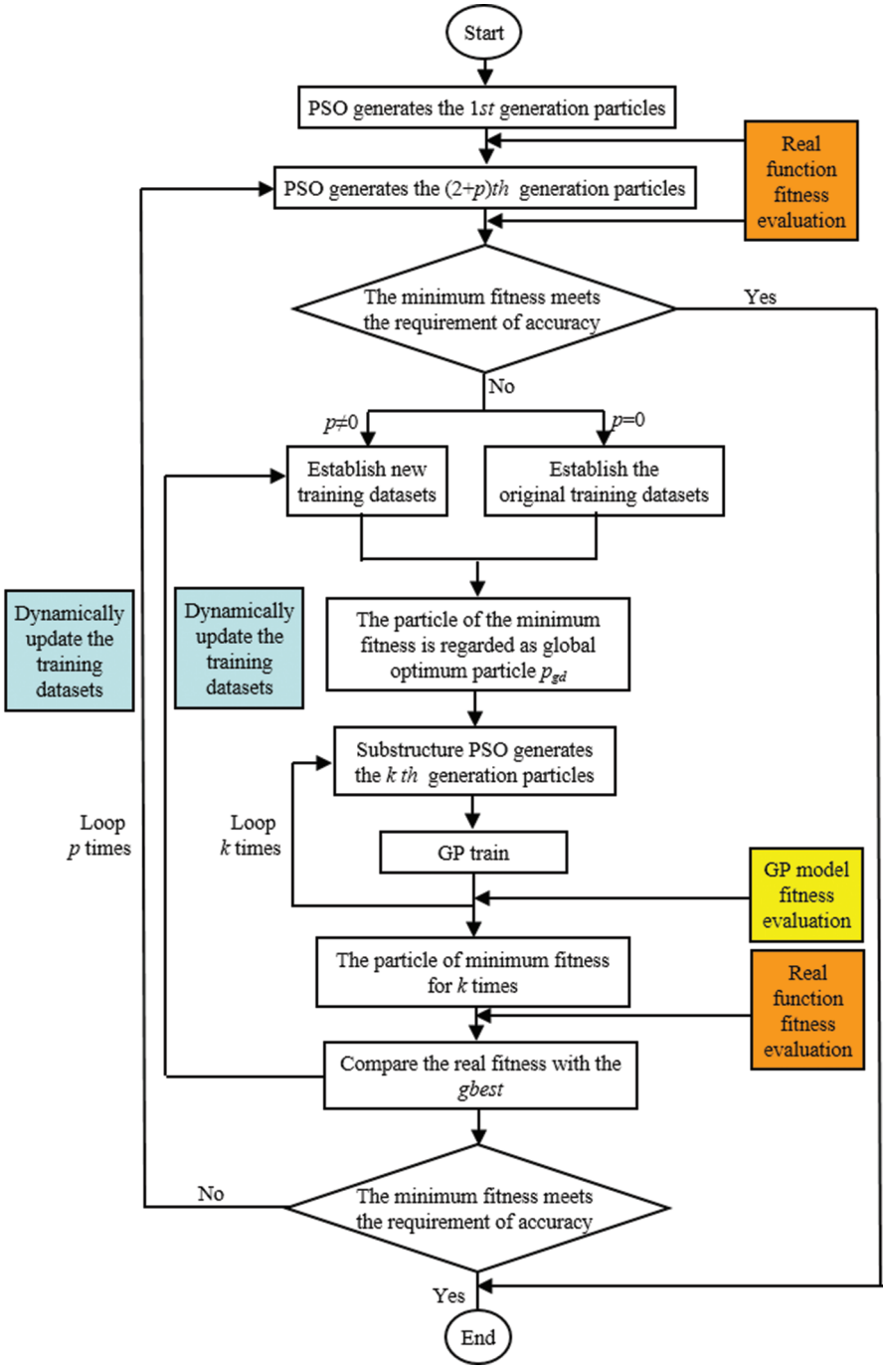


Figure 1 The flow chart of Gaussian process-particle swarm optimization (GP-PSO).

2. Evaluate the fitness of particles of 1st generation by objective function, i.e., real function. Find the optimum particle p_{id} and global optimum particle p_{gd}.
3. Generate N particles of $(2+p)th$ generation using Eq. (1). Evaluate the fitness of these particles using real function.
4. The $(2+p)\times N$ particles are sorted from small fitness to large fitness, and the upper limit $2\times N$ particles and their fitness are selected to establish training datasets.
5. Train GP by the training datasets. Use GP approximate the real function according to Eq. (5).
6. Update the optimum particle p_{id} and global optimum particle p_{gd} at current iteration.
7. Generate $k\times N$ particles using Eq. (1) and evaluate the fitness of N particles using trained GP. Find the best particle with the minimum fitness in k generation, where k can make the best particle more optimizing, k is determined by the complexity of the real function. The more complex the real function is, the greater the value of k will be. Evaluate its fitness using real function and replace its fitness evaluated using trained GP. Compare its fitness to *gbest*, select the smaller one as global optimum particle p_{gd} and update *gbest*.
8. Replace the worst particle with the maximum fitness in training datasets by the global optimum particle and its fitness *gbest*.
9. Repeat steps 3–8 until the stopping criteria is met. For the current implementation the stopping criteria is defined based on the maximum number of iterations reached or accuracy satisfied.

The pseudo code of GP-PSO is given in Figure 2. A MATLAB based program was developed.

```
Begin
    Randomly generate the N particles of 1st generation
    Evaluate particles' fitness using the real function
    Find the optimum particle p_id and global optimum particle p_gd
    p = 0;
  While stopping criteria is not met
    Generate N particles of (2+p) th generation using Eq. (1)
    Evaluate these particles' fitness using real function

    The (2+p)×N particles are sorted from small fitness to large fitness
    The top 2×N particles and their fitness are selected for establishing training datasets

    Train GP by the training datasets
    Update the optimum particle p_id and global optimum particle p_gd at current iteration

     For i=1, k
       Generate N particles using Eq. (1)
       Evaluate the fitness of N particles using trained GP
     End For

     Find the best particle with the minimum fitness in k generation
     Evaluate its fitness using real function and replace its fitness evaluated using trained GP
     Compare its fitness to gbest, select the smaller one as global optimum particle p_gd and update
     gbest
     Replace the worst particle with the maximum fitness in training datasets by this particle and
     its fitness

    p = p+1
  End While
End
```

Figure 2 Pseudo code of Gaussian process-particle swarm optimization (GP-PSO).

4. EXPERIMENTS

4.1. Benchmark Functions

The performance of the GP-PSO algorithm was compared to the conventional PSO algorithm by evaluating convergence velocity and efficiency for benchmark optimization problems. 12 popular benchmark functions are selected in Table 1. It includes 8 unimodal benchmark functions and 4 multimodal benchmark functions. For all functions in Table 1, the minimal function value is $f = 0$ and search space is confined to $[-2, 2]^d$, here d denotes the dimension of the functions.

The Sphere, Ellipsoid, Step and Sumsquares functions are unimodal, convex and differentiable without flat regions. The Cigar, Tablet and Rosen functions have a single global minimum located in a long narrow parabolic shaped flat valley, what's more, it is very hard to find global optimum. Quartic function is unimodal with random noise. Noisy functions are widespread in real-world problems, and every evaluation for the function is disturbed by noise, so the particles' information is inherited and diffused noisily, which makes the problem hard for optimization [49].

Multimodal functions evoke hills and valleys, which are misleading local optima. Schwefel and Ackley have large amount of local minima causing difficulties in finding the global optimum. The Griewangk and Rastrigin are highly multimodal test functions, which are usually used to evaluate the efficiency of optimization algorithm.

For the benchmark functions, we measure the number of the real function evaluation which is required to achieve expected accuracy.

4.2. Parameter Setting

The parameters for PSO include population size (N), lower (lb) and upper (ub) bounds of the search space, maximum number of iterations (t_{max}), inertia parameter (w), maximum velocity of particle (v_{max}) and acceleration constants (c_1 and c_2). For different dimension problem, some parameters remain unchanged: $lb = -2$, $ub = 2$, $v_{max} = 1$, $c_1 = c_2 = 2$. The values of N and t_{max} are listed in Table 2. The maximum number of circular for accelerating local search $k = 10$.

To be fair, for different functions, all algorithms are forced to use the same accuracy of stopping criteria; for different functions, there is different accuracy for stopping criteria which is listed in Table 2. The program for all experiments runs over 30 times independently then the number of function evaluation are averaged finally.

In order to validate the scalability of our algorithm, all functions are tested on 10, 20 and 30 dimensions, respectively [50].

4.3. Parameter Selection

In the optimization, the selection of parameters can significantly influence on the performance of the proposed algorithm. In this section, some essential parameters are discussed, and how they impact on the optimization results are forecasted. Afterward, the recommendations are provided accordingly.

lb and ub are the lower and upper bounds of the search space, respectively. They are distributed symmetrically about the searching value. In this paper the minimal function value is 0, so the search space is confined to $[-2, 2]$. The computational cost will increase with the interval between lb and ub; specifically speaking, if it is too small (i.e., $[-0.1, 0.1]$), any optimization method can easily find the optimal results, that can't be used for evaluating the performance differences between different methods. In this paper, the aim for this benchmark optimization problem is to compare the performance of the GP-PSO algorithm and PSO algorithm. As long as the performance of the two algorithms can be compared under the same condition, the smaller interval is used to save the computational time. Therefore, the bound of $[-2, 2]$ is big enough to compare the performance of the GP-PSO algorithm and PSO algorithm.

Table 1 Parameter settings of the benchmark functions.

Function	Formulation	Trait	Search Range
Sphere	$f_1(x) = \sum_{i=1}^{n} x_i^2$	Unimodal	$[-2,2]^d$
Ellipsoid	$f_2(x) = \sum_{i=1}^{n} i x_i^2$	Unimodal	$[-2,2]^d$
Step	$f_3(x) = \sum_{i=1}^{n} (x_i + 0.5)^2$	Unimodal	$[-2,2]^d$
Sumsquares	$f_4(x) = \sum_{i=1}^{n} i x_i^2$	Unimodal	$[-2,2]^d$
Cigar	$f_5(x) = x_1^2 + 10\sum_{i=2}^{n} x_i^2$	Unimodal	$[-2,2]^d$
Tablet	$f_6(x) = 10^6 x_1^2 + \sum_{i=2}^{n} x_i^2$	Unimodal	$[-2,2]^d$
Rosen	$f_7(x) = \sum_{i=1}^{n-1} \left(100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right)$	Unimodal	$[-2,2]^d$
Quartic	$f_8(x) = \sum_{i=1}^{n} i x_i^4 + random(0,1)$	Unimodal	$[-2,2]^d$
Ackley	$f_9(x) = 20 + e - 20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)$	Multimodal	$[-2,2]^d$
Schwefel	$f_{10}(x) = 418.9829n - \sum_{i=1}^{n} x_i \sin\left(\sqrt{\lvert x_i\rvert}\right)$	Multimodal	$[-2,2]^d$
Griewank	$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	Multimodal	$[-2,2]^d$
Rastrigin	$f_{12}(x) = \sum_{i=1}^{n}\left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$	Multimodal	$[-2,2]^d$

Table 2 Parameter settings of GP-PSO and PSO.

Function	10 Dimensions			20 Dimensions			30 Dimensions		
	N	t_{max}	Accuracy	N	t_{max}	Accuracy	N	t_{max}	Accuracy
Sphere	30	2000	1.00E-03	30	8000	1.00E-03	50	20000	1.00E-03
Ellipsoid	30	2000	1.00E-03	30	8000	1.00E-03	50	20000	1.00E-03
Step	30	2000	1.00E-03	30	8000	1.00E-03	50	20000	1.00E-03
Sumsquares	30	2000	1.00E-03	30	8000	1.00E-03	50	20000	1.00E-03
Cigar	30	2000	1.00E-03	30	8000	1.00E-03	50	20000	1.00E-03
Tablet	30	2000	1.00E-03	30	8000	1.00E-03	50	20000	1.00E-03
Rosen	30	10000	3.00E+01	30	20000	3.00E+01	50	50000	3.00E+01
Quartic	30	10000	1.00E-01	30	20000	1.00E-01	50	50000	1.00E-01
Ackley	30	2000	1.00E-03	30	8000	1.00E-03	50	20000	1.00E-03
Schwefel	30	2000	1.00E-03	30	8000	1.00E-03	50	20000	1.00E-03
Griewank	30	2000	1.00E-03	30	8000	1.00E-03	50	20000	1.00E-03
Rastrigin	30	10000	3.00E+01	30	20000	3.00E+01	50	50000	3.00E+01

GP, Gaussian process; PSO, particle swarm optimization.

$\boldsymbol{v}_{\max}$ is a parameter specified by the user, then the velocity on that dimension is limited to $\boldsymbol{v}_{\max}$. $\boldsymbol{v}_{\max}$ determines the resolution of the search regions between the present and target position. The PSO works well if $\boldsymbol{v}_{\max}$ is set to the value of the dynamic range of each variable (on each dimension). The dynamic range is [−2, 2] in this paper, so we choose the $\boldsymbol{v}_{\max} = 1$.

$c1$ and $c2$ are acceleration constants usually defined as $c1 = c2 = [1.8, 2.0]$, that can be found in literature [41]. Any value in this interval could be chosen. In this paper, we choose c1 = c2 = 2.0.

N is usually determined by the difficulty of the optimization problem. Small population size results in unreliable results, while the large size will increase computational cost and may cause slow convergence. The population size should be selected with the consideration of the tradeoff between the solution quality and the computation time. It is always determined by the experience. In this paper, we choose the N based on the following reference [51]. In this reference, 10, 30, 50, 80 and 100 population sizes are used to solve the optimization problem, respectively. In our work, the purpose is to evaluate the performance of two algorithms. Therefore, in order to find a balance between accuracy and computational cost, the value is selected as 30 and 50 depending on the complexity of the optimization problem.

t_{max} determines the maximum run times of algorithm. Because all the different algorithms are forced to use the same accuracy as the termination criterion, the t_{max} should be big enough to make the algorithms meet the termination criterion. There is no certain limit of this parameter. For example, the Sphere function is relatively simple, so 2000 is sufficient; meanwhile, the Rastrigin function is more complex, 50000 is adapted.

k represents the increment of loop as substructure of PSO algorithm. In GP-PSO, the GP is used as an inexpensive fitness function surrogate and a powerful tool to predict the global optimum solution for accelerating the local search of PSO. The fitness of total k generations doesn't need to be calculated; instead, it is predicted by GP model. Usually, the prediction of GP approximates to the optimal solution by using larger k. However, larger k also causes high computational cost. Therefore, $k = 10$ in our paper is determined, which can satisfy our optimal requirement. In conclusion, all of the parameters mentioned are summarized in Table 3.

Table 3 The summation of parameter selection.

Parameter	Influence	Recommendation
lb	Computational cost	−2
ub	Computational cost	2
v_{max}	Convergence	1
c_1	Accuracy, computational cost and convergence	2
c_2	Accuracy, computational cost and convergence	2
N	Accuracy, computational cost and convergence	30 or 50
t_{max}	Convergence	2000–50000
k	Computational cost and convergence	10

4.4. Results Analysis

The total number of the real function evaluation using GP-PSO and PSO are shown in Table 4, respectively. It can be seen from Table 4, total number of real function evaluation in GP-PSO is much less than that PSO requires.

For 10-, 20-, 30-dimensional benchmark functions, the average multiple of the real function evaluation is about 12, 24, 18 times, respectively, the minimum multiple of it is 3 times (Rastrigin function with 10 dimension), and the maximum multiple of the real function evaluation is 132 times (Rastrigin function with 20 dimension).

In order to evaluate the performance of the proposed algorithm, the Rastrigin and Rosen function with 30 dimensions was used due to their high difficulty in searching optimal value. Usually, the mean value and corresponding standard deviation are two important features to evaluate the accuracy and robustness of algorithm. The GP-PSO was compared with PSO, PSO-w-local [52] and PSO-cf-local [53], respectively. The results are summarized in the Table 5. Except

Table 4 Comparison of the number of function evaluation between GP-PSO and PSO.

Function	10 Dimensions			20 Dimensions			30 Dimensions		
	PSO	GP-PSO	Multiple	PSO	GP-PSO	Multiple	PSO	GP-PSO	Multiple
Sphere	22230	1549	14	92550	9299	10	427400	32129	13
Ellipsoid	25140	2572	10	106440	12306	9	435100	48704	9
Step	24360	2014	12	94800	7439	13	388900	33659	12
Sumsquares	24180	2200	11	95280	8524	11	440750	52733	8
Cigar	29610	4680	6	118650	16398	7	476850	65075	7
Tablet	24180	2076	12	98130	8276	12	426250	41462	10
Rosen	1260	154	8	110880	2975	37	816150	41819	20
Quartic	20070	743	27	139050	4370	32	847100	28610	30
Ackley	29580	4215	7	112470	14817	8	470900	56609	8
Schwefel	23520	4184	6	101610	17886	6	471850	62474	8
Griewank	16950	774	22	73620	4711	16	343900	19175	18
Rastrigin	810	278	3	114780	867	132	802050	10862	74

GP, Gaussian process; PSO, particle swarm optimization.

Table 5 Comparison results of GP-PSO with other algorithms for 30-dimensional problems.

Algorithm	Rastrigin				Rosen			
	Best	Worst	Mean	Std	Best	Worst	Mean	Std
PSO	2.97E+01	3.00E+01	2.99E+01	1.90E-01	2.93E+01	3.00E+01	2.96E+01	3.10E-01
PSO-*w*-local [52]	2.09E+01	3.38E+01	2.64E+01	4.08E+00	2.41E+01	2.94E+01	2.60E+01	2.26E+00
PSO-cf-local [53]	2.89E+01	6.17E+01	4.03E+01	1.20E+01	1.77E+01	2.32E+01	1.95E+01	2.44E+00
GP-PSO	2.94E+00	3.00E+01	2.97E+01	8.00E-02	2.91E+01	3.00E+01	2.95E+01	2.80E-01

GP, Gaussian process; PSO, particle swarm optimization.

for PSO-cf-local, the mean value calculated from the other three algorithms approach to the actual value of 30 for both Rastrigin and Rosen function. The PSO-cf-local has the lowest accuracy in our case. The proposed GP-PSO can achieve the same accuracy of PSO and PSO-w-local. However, it has the lowest standard deviation. That means the proposed GP-PSO algorithm has better performance in robustness in the premise of sufficient accuracy.

In order to give a visualized and straightforward comparison between the proposed GP-PSO and PSO, the comparison of the performances of GP-PSO and PSO for 30-dimensional benchmark functions are shown in Figure 3. In the figures, the convergence trends of variants of PSO and GP-PSO in a random run can be identified. These figures show that how GP-PSO converged toward the optimal solution faster than PSO algorithm. It indicates that the proposed GP-PSO is much more efficient than PSO to achieve identical accuracy.

In addition, both GP-PSO and PSO can achieve expected accuracy (see Table 2) after over 30 times runs for all experiments. It demonstrates that the robustness of GP-PSO is the same as PSO. The robustness of two algorithms is comparable, because the GP-PSO does not change the principle rule of evolution; in our algorithm, we just take advantage of GP to estimate the most potential solution then improve the searching speed of PSO. GP-PSO and PSO have the ability of searching the global optimum, but GP-PSO is more efficient than PSO.

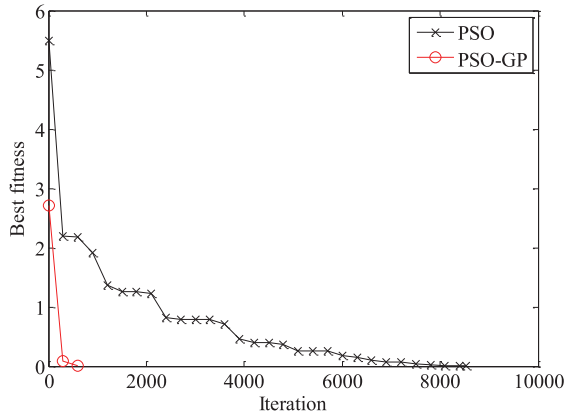
As shown in Figure 3, the converge of GP-PSO is the fastest within iteration of 1000. Therefore, the converge of PSO and GP-PSO with iteration of 300/600/900 in the different 30-dimensional functions were compared in Table 6. The PSO and GP-PSO start to show distinct performance in optimization from iteration of 300. With the increase of iteration, the optimal value decreases for both methods. For iteration of 900, PSO can't meet the requirement of accuracy for all of the functions. GP-PSO associated with Rastrigin function will achieve the accuracy with iteration of 300; for Griewank and Sphere, Step, Tablet, Quartic function, iteration of 600 and 900 are sufficient, respectively. The iteration of residual function is over 900 for both methods.

5. CONCLUSIONS AND FUTURE STUDY

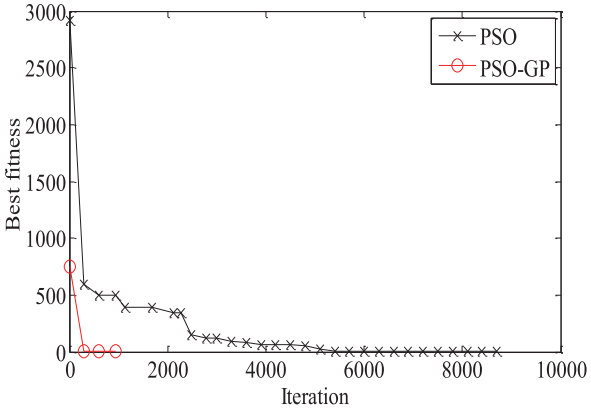
It is important to reduce the number of function evaluations when optimizing expensive fitness functions. This can be achieved by exploiting knowledge of past evaluated points to train an empirical model, which can be used as inexpensive surrogate of the fitness function. This paper has shown that the combination of PSO and GP can be used as fitness function surrogates to solve

the complex problems of function global optimization. In the proposed algorithm, the GP dramatically reduces the number of function evaluation and provides accurate estimation of the global optimum solution; in addition, the dynamically update training datasets for training GP improves the approximation accuracy of the GP. The results have clearly demonstrated that the proposed algorithm is superior to PSO, which is widely accepted as an efficient tool to solve the very ill-conditioned problems of global optimization. Excellent performance and easy implementation of GP-PSO shows a good potential in computationally expensive optimization problems.

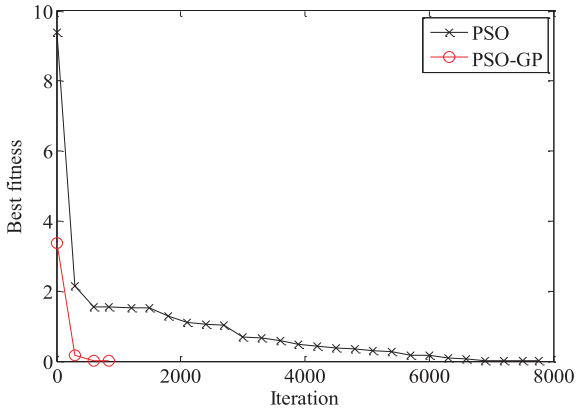
The proposed algorithm is specified for the complex and nonlinear optimization problem in many engineering fields, such as mechanical, civil and aerospace engineering. In most of the cases, the computational cost is extremely high, which can't be afforded by means of conventional algorithm; especially for large-scale finite element analysis or computational fluid dynamics simulations. Furthermore, from the aspect of complexity, it is challenging because many global optimization cases involve objective functions of multimodal, highly nonlinear, with steep and flat regions and irregularities. In this work, though the applicability can't be fully verified, the superiority of proposed algorithm has been proved by



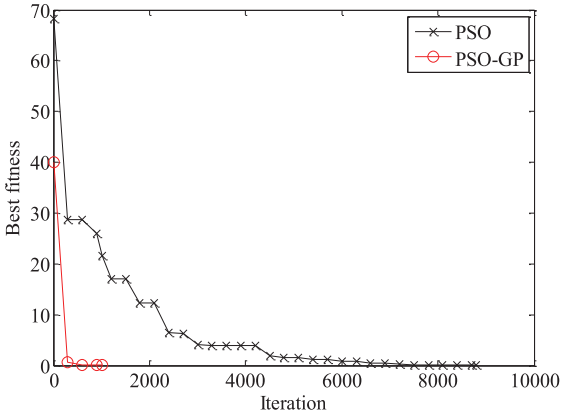
(**a**) Sphere function



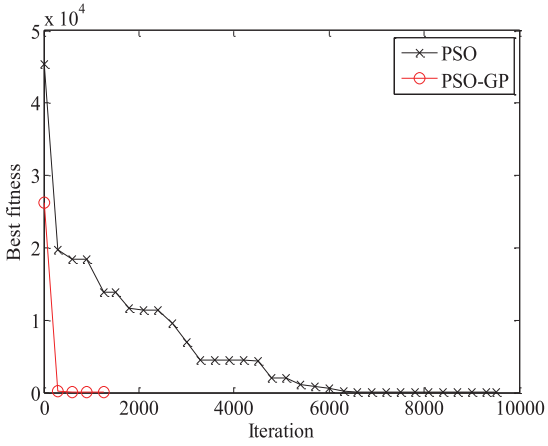
(b) Ellipsoid function



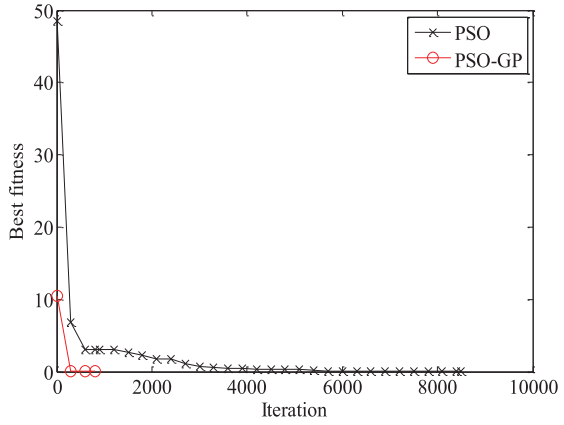
(**c**) Step function



(d) Sumsquares function



(**e**) Cigar function



(**f**) Tablet function

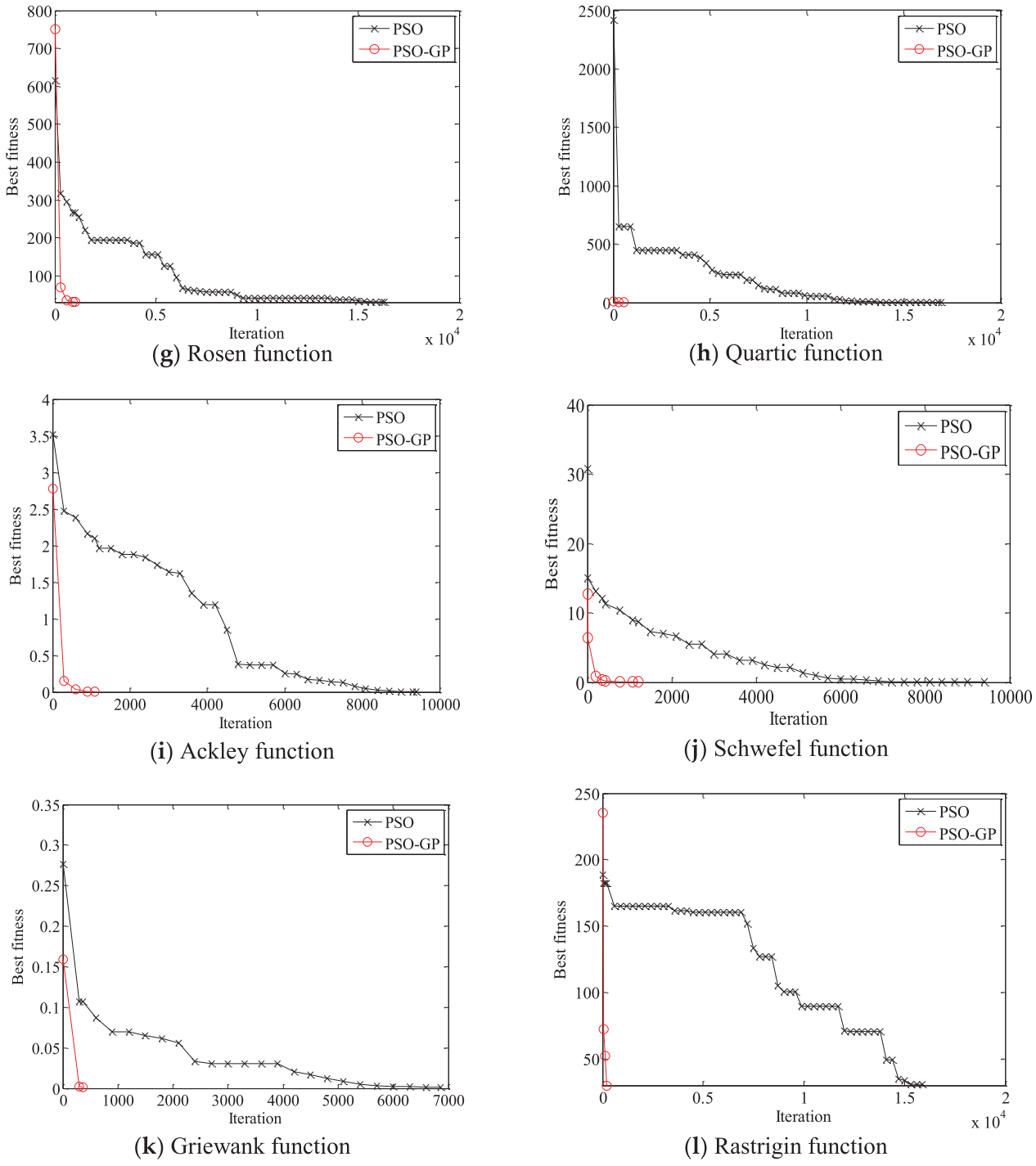


Figure 3 Convergence of the Gaussian process-particle swarm optimization (GP-PSO) algorithm for 30-dimensional benchmark functions.

comparing with variety of typical standard benchmarks. Moreover, the GP in this algorithm is only based on exponential covariance function. In the future, more covariance functions will be tested; such as Matern covariance function and linear covariance function. Additionally, the performance of the proposed algorithm is only tested with the typical benchmark functions in this work. The expansibility and applicability will be validated by introducing the data from the practical cases. And also, the proposed strategy will be extended in terms of more combinations to form up the new algorithm.

CONFLICT OF INTEREST

The authors declare no conflicts of interest.

AUTHORS' CONTRIBUTIONS

Yan Zhang and Hongyu Li proposed the main idea of the paper, developed and validated the algorithm, and analyzed the results. Enhe Bao and Aiping Yu contributed with the idea to provide the

Table 6 Comparison of the fitness evaluation between GP-PSO and PSO.

Function	300		600		900	
	PSO	GP-PSO	PSO	GP-PSO	PSO	GP-PSO
Sphere	2.198442	0.087529	2.175041	0.001023	1.911124	0.000913
Ellipsoid	594.628497	2.414306	493.550394	0.156631	493.550394	0.002018
Step	2.155441	0.176797	1.557810	0.009128	1.523526	0.000709
Sumsquares	28.746118	0.587226	28.746118	0.098386	25.947067	0.003208
Cigar	19763.982314	247.688017	18366.828056	5.043769	18366.828056	0.251856
Tablet	6.823563	0.071880	3.118476	0.006294	3.118476	0.000822
Rosen	317.017070	69.640895	295.183256	35.633361	266.358637	30.845138
Quartic	651.919176	0.174035	651.919176	0.086739	651.919176	0.000736
Ackley	2.473627	0.149701	2.381386	0.032678	2.165373	0.006058
Schwefel	12.146868	0.414506	10.484639	0.076709	8.918804	0.009359
Griewank	0.107419	0.002093	0.087207	0.000665	0.069805	0.000665
Rastrigin	182.095993	29.991134	164.769693	29.991134	164.769693	29.991134

GP, Gaussian process; PSO, particle swarm optimization.

benchmark of the algorithm. Lu Zhang conducted the literature survey and contributed to the introduction section and polished the paper. Yan Zhang, Hongyu Li and Lu Zhang wrote the manuscripts with the support from other co-authors.

ACKNOWLEDGMENT

The present work was supported by the National Natural Science Foundation of China (Grant Nos. 51708147, 51409051 and 51568014), the Natural Science Foundation of Guangxi (Grant Nos. 2017GXNSFBA198184 and 2014GXNSFBA118256), Guangxi Science and Technology Base and Special Fund for Talents Program (Grant No. Guike AD19110044) and Guangxi Innovation Driven Development Project (Science and Technology Major Project, Grant No. Guike AA18118008). The authors also would like to thank the partial support from Guangxi Key Laboratory of Geomechanics and Geotechnical Engineering (Grant Nos. 14-B-01 and 2015-A-01-05).

REFERENCES

[1] A.R.M. Rao, K. Sivasubramanian, Multi-objective optimal design of fuzzy logic controller using a self configurable swarm intelligence algorithm, Comput. Struct. 86 (2008), 2141–2154.
[2] A.H. Gandomi, A.R. Kashani, D.A. Roke, M. Mousavi, Optimization of retaining wall design using recent swarm intelligence techniques, Eng. Struct. 103 (2015), 72–84.
[3] E. Bonabeau, D.D.R.D.F. Marco, M. Dorigo, G. Theraulaz, Swarm Intelligence: From Natural to Artificial Systems, 1999.
[4] J. Liao, L. Tang, G. Shao, Q. Qiu, C. Wang, S. Zheng, X. Su, A neighbor decay cellular automata approach for simulating urban expansion based on particle swarm intelligence, Int. J. Geogr. Inf. Sci. 28 (2014), 720–738.
[5] Z. Cui, B. Sun, G. Wang, Y. Xue, J. Chen, A novel oriented cuckoo search algorithm to improve DV-Hop performance for cyber-physical systems, J. Parallel Distr. Com. 103 (2017), 42–52.
[6] M. Zhang, H. Wang, Z. Cui, J. Chen, Hybrid multi-objective cuckoo search with dynamical local search, Memet. Comput. 10 (2018), 199–208.
[7] X. Cai, X. Gao, Y. Xue, Improved bat algorithm with optimal forage strategy and random disturbance strategy, Int. J. Bio-Inspir. Com. 8 (2016), 205–214.
[8] X. Cai, H. Wang, Z. Cui, J. Cai, Y. Xue, L. Wang, Bat algorithm with triangle-flipping strategy for numerical optimization, Int. J. Mach. Learn. Cyb. 9 (2018), 199–215.
[9] M. Celik, F. Koylu, D. Karaboga, CoABCMiner: an algorithm for cooperative rule classification system based on artificial bee colony, Int. J. Artif. Intell. T. 25 (2016), 1550028.
[10] H.M. Jiang, K. Xie, Y.F. Wang, Optimization of pump parameters for gain flattened Raman fiber amplifiers based on artificial fish school algorithm, Opt. Commun. 284 (2011), 5480–5483.
[11] M. Dorigo, V. Maniezzo, A. Colorni, Ant system: optimization by a colony of cooperating agents, IEEE Trans. Syst. Man. Cybern. Part B-Cybern. 26 (1996), 29–41.
[12] R. Eberhart, J. Kennedy, New optimizer using particle swarm theory, in MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, 1995, pp. 39–43.
[13] M.M. Ali, P. Kaelo, Improved particle swarm algorithms for global optimization, Appl. Math. Comput. 196 (2008), 578–593.
[14] M.M. Noel, A new gradient based particle swarm optimization algorithm for accurate computation of global minimum, Appl. Soft Comput. 12 (2012), 353–359.
[15] M. Sopa, N. Angkawisittpan, An application of cuckoo search algorithm for series system with cost and multiple choices constraints, Procedia Comput. Sci. 86 (2016), 453–456.
[16] T. Gaber, S. Abdelwahab, M. Elhoseny, A.E. Hassanien, Trust-based secure clustering in WSN-based intelligent transportation systems. Comput. Netw. 146 (2018), 151–158.
[17] J. Kennedy, R. Eberhart, Particle swarm optimization (PSO), in Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ, 1995, pp. 1942–1948.
[18] D.E. Goldberg, GA in Search, Optimization, and Machine Learning, Addison-Wesley, Boston, MA, 1989.
[19] R. Hassan, B. Cohanim, O. De Weck, G. Venter, A comparison of particle swarm optimization and the genetic algorithm, in 46th AIAA/ASME/ASCE/AHS/ASC structures, structural dynamics and materials conference, Austin, TX, 2005, pp. 1897.

[20] D. Wang, Z. Wu, Y. Fei, W. Zhang, Structural design employing a sequential approximation optimization approach, Comput. Struct. 134 (2014), 75–87.
[21] N.H. Awad, M.Z. Ali, R. Mallipeddi, P.N. Suganthan, An improved differential evolution algorithm using efficient adapted surrogate model for numerical optimization, Inf. Sci. 451 (2018), 326–347.
[22] G. Su, Q. Jiang, A cooperative optimization algorithm based on gaussian process and particle swarm optimization for optimizing expensive problems, in 2009 International Joint Conference on Computational Sciences and Optimization, IEEE, Hainan, Sanya, China, 2009, Vol. 2, pp. 929–933.
[23] K. Mistry, L. Zhang, S.C. Neoh, C.P. Lim, B. Fielding, A micro-GA embedded PSO feature selection approach to intelligent facial emotion recognition, IEEE Trans. Cybern. 47 (2016), 1496–1509.
[24] H. Garg, A hybrid GSA-GA algorithm for constrained optimization problems, Inf. Sci. 478 (2019), 499–523.
[25] L.Y. Chuang, H.W. Chang, C.J. Tu, C.H. Yang, Improved binary PSO for feature selection using gene expression data, Comput. Biol. Chem. 32, (2008), 29–38.
[26] C. Praveen, R. Duvigneau, Low cost PSO using metamodels and inexact pre-evaluation: application to aerodynamic shape design, Comput. Methods Appl. Mech. Eng. 198 (2009), 1087–1096.
[27] S. Selleri, M. Mussetta, P. Pirinoli, R.E. Zichm, L. Matekovits, Differentiated meta-PSO methods for array optimization, IEEE Trans. Antennas Propag. 56 (2008), 67–75.
[28] M.V.J.J. Suresh, K.S. Reddy, A.K. Kolar, ANN-GA based optimization of a high ash coal-fired supercritical power plant, Appl. Energ. 88 (2011), 4867–4873.
[29] A. Ratle, Accelerating the convergence of evolutionary algorithms by fitness landscape approximation, in International Conference on Parallel Problem Solving from Nature, Springer, Berlin, Germany, 1998, pp. 87–96.
[30] K.C. Giannakoglou, Optimization and inverse design in aeronautics: how to couple genetic algorithms with radial basis function networks, in: J. Periaux, P. Joly, O. Pironneau, E. Onate (Eds.), Innovative Tools for Scientific Computation in Aeronautical Engineering, CIMNE, Barcelona, Spain, 2001.
[31] Y. Jin, A comprehensive survey of fitness approximation in evolutionary computation, Soft Comput. 9 (2005), 3–12.
[32] Y. Jin, Surrogate-assisted evolutionary computation: recent advances and future challenges, Swarm Evol. Comput. 1 (2011), 61–70.
[33] J. Kacprzyk, Advances in Soft Computing, Heidelberg, 2001.
[34] Y. Zhang, Y. Jun, G. Wei, L. Wu, Find multi-objective paths in stochastic networks via chaotic immune PSO, Expert Syst. Appl. 37 (2010), 1911–1919.
[35] M.T.M. Emmerich, K.C. Giannakoglou, B. Naujoks, Single-and multiobjective evolutionary optimization assisted by Gaussian random field metamodels, IEEE Trans. Evol. Comput. 10 (2006), 421–439.
[36] J. Knowles, ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems, IEEE Trans. Evolut. Comput. 10 (2006), 50–66.
[37] Q. Zhang, W. Liu, E. Tsang, B. Virginas, Expensive multiobjective optimization by MOEA/D with Gaussian process model, IEEE Trans. Evol. Comput. 14 (2009), 456–474.
[38] N. Higashi, H. Iba, Particle swarm optimization with Gaussian mutation, in Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No.03EX706), Indianapolis, IN, 2003, pp. 72–79.
[39] R.A. Krohling, Gaussian Particle Swarm with jumps, in 2005 IEEE Congress on Evolutionary Computation, Edinburgh, Scotland, UK, 2005, pp. 1226–1231.
[40] Y. Zhang, B. Liu, Y. Hou, Z. Zeng, An intelligent back analysis optimization method of constitutive parameters for surrounding rock, Unsaturated soil mechanics - from theory to practice, in Proceedings of the 6th Asia Pacific Conference on Unsaturated Soils, Guilin, China, 2015, pp. 601–605.
[41] G. Wang, Z. Ma, Hybrid particle swarm optimization for first-order reliability method, Comput. Geotech. 81 (2017), 49–58.
[42] J. Kennedy, The behavior of particles, in: V.W. Porto, N. Saravanan, D. Waagen, A.E. Eiben (Eds.), Evolutionary Programming VII, Springer, Berlin, Heidelberg, 1998, pp. 581–589.
[43] D.J.C. MacKay, Introduction to Gaussian processes, in: C.M. Bishop (Ed.), Neural Networks and Machine Learning, NATO ASI Series, Springer, Berlin, Germany, 1998, pp. 133–166.
[44] G. Su, L. Yan, Y. Song, Gaussian process for non-linear displacement time series prediction of landslide, J. China Univ. Geosci. 18 (2007), 219–221.
[45] J. Hensman, R. Mills, S.G. Pierce, K. Worden, M. Eaton, Locating acoustic emission sources in complex structures using Gaussian processes, Mech. Syst. Signal Proc. 24 (2010), 211–223.
[46] M. Pal, S. Deswal, Modelling pile capacity using Gaussian process regression, Comput. Geotech. 37 (2010), 942–947.
[47] C.K.I. Williams, C.E. Rasmussen, Gaussian Processes for Machine Learning, Cambridge, MA, 2006.
[48] G. Su, Gaussian process assisted differential evolution algorithm for computationally expensive optimization problems, in 2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application, Wuhan, China, 2008, pp. 272–276.
[49] Y. Shi, H. Liu, L. Gao, G. Zhang, Cellular particle swarm optimization, Inf. Sci. 181 (2011), 4460–4493.
[50] Y. Wang, B. Li, T. Weise, J. Wang, B. Yuan, Q. Tian, Self-adaptive learning based particle swarm optimization, Inf. Sci. 181 (2011), 4515–4538.
[51] W. Zhang, Y. Liu, Reactive power optimization based on PSO in a practical power system, in IEEE Power Engineering Society General Meeting, Denver, CO, 2004, pp. 239–243.
[52] Y. Shi, R.C. Eberhart, A modified particle swarm optimizer, in Proceedings IEEE Conference on Evolutionary Computation, Anchorage, AK, 1998.
[53] J. Kennedy, R. Mendes, Population structure and particle swarm performance, in Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600), Honolulu, HI, 2002, pp.1671–1676.