J Glob Optim (2017) 68:641–662
DOI 10.1007/s10898-016-0484-7

Pseudo expected improvement criterion for parallel EGO algorithm

Dawei Zhan[1] · **Jiachang Qian**[1] · **Yuansheng Cheng**[1]

Received: 19 October 2015 / Accepted: 21 November 2016 / Published online: 29 November 2016

Abstract The efficient global optimization (EGO) algorithm is famous for its high efficiency in solving computationally expensive optimization problems. However, the expected improvement (EI) criterion used for picking up candidate points in the EGO process produces only one design point per optimization cycle, which is time-wasting when parallel computing can be used. In this work, a new criterion called pseudo expected improvement (PEI) is proposed for developing parallel EGO algorithms. In each cycle, the first updating point is selected by the initial EI function. After that, the PEI function is built to approximate the real updated EI function by multiplying the initial EI function by an influence function of the updating point. The influence function is designed to simulate the impact that the updating point will have on the EI function, and is only corresponding to the position of the updating point (not the function value of the updating point). Therefore, the next updating point can be identified by maximizing the PEI function without evaluating the first updating point. As the sequential process goes on, a desired number of updating points can be selected by the PEI criterion within one optimization cycle. The efficiency of the proposed PEI criterion is validated by six benchmarks with dimension from 2 to 6. The results show that the proposed PEI algorithm performs significantly better than the standard EGO algorithm, and gains significant improvements over five of the six test problems compared against a state-of-the-art parallel EGO algorithm. Furthermore, additional experiments show that it affects the convergence of the proposed algorithm significantly when the global maximum of the PEI function is not found. It is recommended to use as much evaluations as one can afford to find the global maximum of the PEI function.

✉ Yuansheng Cheng
yscheng@hust.edu.cn

Dawei Zhan
zhandawei@hust.edu.cn

Jiachang Qian
qianjiachang@163.com

[1] School of Naval Architecture and Ocean Engineering, Huazhong University of Science and Technology, Wuhan 430074, China

Keywords Efficient global optimization · Expected improvement · Parallel computing · Pseudo expected improvement · Influence function

1 Introduction

In today's engineering industry, designers often need to optimize systems that involve computationally expensive models such as finite element analysis (FEA) models and computational fluid dynamics (CFD) models. Many derivative-free algorithms have been developed for solving the optimization work of these black-box problems [1,2]. Among them, the population-based methods such as genetic algorithm (GA) and particle swarm optimization (PSO) algorithm are reported to be very reliable in finding the global optimum of a black-box problem. However, they often need thousands of evaluations of the objective function, which may be more than one can afford for a computationally expensive problem.

In the past two decades, surrogate-based optimization methods have attracted more and more attention in solving computationally expensive optimization problems [3,4]. Using computationally inexpensive surrogate models to replace the computationally expensive models in the optimization process can reduce the number of evaluations of the computationally expensive models, thus speed up the optimization process significantly. The efficient global optimization (EGO) algorithm is one of the most successful surrogate-based optimization methods [5,6]. The EGO algorithm begins by fitting a Kriging model with a set of initial designs, then proceeds iteratively. At each iteration, one candidate point is selected by maximizing an infill sampling criterion called expected improvement (EI) and is then evaluated to refit the Kriging model. The process stops until a stopping criterion is met. The EGO algorithm has been widely studied and extended to deal with constrained problems [7,8], noisy problems [9,10] and multi-objective problems [11,12].

One of the main limitations of the EGO algorithm is that it is not suitable for parallelization due to the point-to-point nature of the standard EI criterion. Different ideas have been employed to extend the traditional EGO algorithm to yield multiple candidate points within one optimization cycle. The evaluations associated with these candidate points can be distributed on different processors. Therefore, the wall clock time (the time of total cycles rather than the time of total evaluations) can be reduced. The strategies for developing parallel EGO algorithms are reviewed in this work based on how many optimizations and how many criteria are used to produce q ($q > 1$) candidate points at each iteration.

The first group of strategies uses one optimization on one criterion to produce q candidate points in each cycle. Ginsbourger et al. [13] focused on deriving the q-point expected improvement (q-EI) criterion which calculates the expected improvement value when q points are added to the design sets at the same time. An analytic expression was derived for 2-EI criterion, and Monte Carlo simulations were used to calculate the q-EI for general cases. The q-EI criterion only needs to be optimized one time to produce q candidate points in each cycle, but the dimension of the optimization problem increases to $q \times d$ (d is the dimension of the design variables). Besides, the evaluations of the q-EI criterion by Monte Carlo simulations can be time-consuming. A multi-modal optimization approach [14] and a multi-objective optimization approach [15]were used for producing q candidate designs in each cycle. The multi-modal optimization approach [14] tries to find all the local maxima of the EI function and selects the q best maxima as candidate points. The multi-objective optimization approach [15] solves a bi-objective problem (the two objectives are the two terms of the EI function) and selects q candidates from the obtained Pareto solutions. The advantage of these two approaches is that q candidate points can be obtained by using only one optimization in each cycle. The shortcoming of the multi-modal optimization approach

is that the number of the local maxima of the EI function is uncertain (sometimes is less than q), which makes this approach not very suitable to parallel computation [16]. How to select q candidate points from a large amount of Pareto solutions is the challenge of the multi-objective optimization approach.

The second group of strategies optimizes one criterion q times in each cycle to get q candidate designs. As opposite to approaches in the first group which select q points simultaneously, these approaches select q candidate points sequentially within one optimization cycle. After selecting the first updating point based on the standard EI function, instead of evaluating the selected updating point to update the Kriging model, Ginsbourger et al. [13] used a 'fake' value to update the Kriging model for producing the next candidate point. As the process goes on, q candidate points can be identified sequentially without doing expensive evaluations within each optimization cycle. This approach is called *Kriging Believer* or *Constant Liar* when the 'fake' value is set to the Kriging prediction value or a constant value. Although the approaches in the second group need to optimize the EI criterion q times to get q candidate points, these optimizations are all carried on the inexpensive EI functions, thus will not cost much time.

The third group uses one optimization on multiple criteria (including other criteria besides the EI criterion) to produce q updating points in each cycle. These approaches often use a multi-objective optimization approach [17], which have the same mechanism of producing multiple candidates as the multi-objective approach in the first group [15].

The approaches in the last group get q updating points from q different criteria. The q different criteria which are optimized by q optimizers could be q different kinds of criteria [18], or one kind of criterion with q different settings [19], or one kind of criterion from q different surrogates [20]. The benefit of these approaches is that the q optimizations can be executed in parallel on q processors, but the number of updating points is often restricted by the number of criteria (or surrogates) in these approaches.

In this paper, a new infill criterion called pseudo expected improvement (PEI) is proposed to produce multiple updating points for parallel computing. The proposed PEI criterion tries to approximate the real updated EI function by multiplying the initial EI function by the influence function (IF) of the updating point, thus is able to sequentially select multiple candidate points per cycle without evaluating the expensive functions. A new parallel EGO algorithm called EGO–PEI algorithm is proposed based on the PEI criterion in this work. The EGO–PEI algorithm is easy to understand and implement. To study its efficiency, the proposed EGO–PEI algorithm is compared against the standard EGO algorithm and the *Constant Liar* algorithm over six analytical functions with dimension from 2 to 6.

The remainder of the paper is organized as follows. Section 2 briefly presents the standard efficient global optimization algorithm. Section 3 introduces the proposed pseudo expected improvement (PEI) criterion and the EGO–PEI algorithm in detail. Sections 4 and 5 present the numerical experiments and the discussion of the results, respectively. Conclusions are given in Sect. 6.

2 Efficient global optimization algorithm

2.1 Kriging model

The Kriging model was first introduced in geology to approximate the distribution of mineral. After Sacks [21] applied it to approximate computer experiments, the Kriging model became

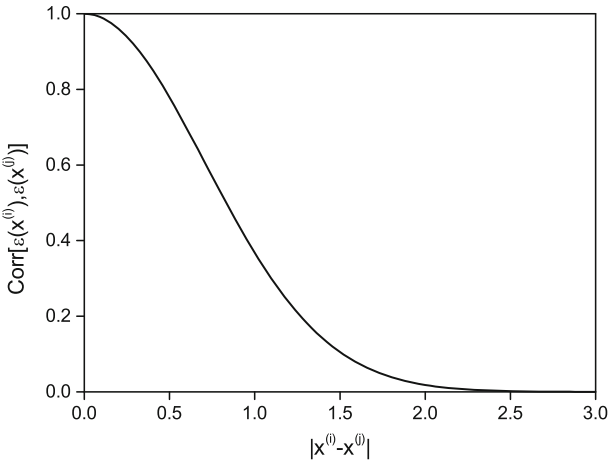

Fig. 1 The correlation function corresponding to $\theta = 1$ and $p = 2$

famous in engineering design and optimization. Only a brief introduction to the Kriging model is given here, detailed derivation of the Kriging model can be seen in [21].

The Kriging model treats the target function as a realization of a Gaussian process. Based on a set of design points $\mathbf{X} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n)}\}$ and their corresponding function values $\mathbf{Y} = \{y^{(1)}, y^{(2)}, \ldots, y^{(n)}\}$, a simple Kriging model is built as:

$$y(\mathbf{x}) = \mu + \varepsilon(\mathbf{x}), \tag{1}$$

where μ is the mean of the Gaussian process, and $\varepsilon(\mathbf{x})$ is the error term which is normally distributed with mean zero and variance σ^2. Most regression models assume that the error terms between two points $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ are independent, which is blatantly false when modeling a deterministic computer model [5]. On the contrary, the Kriging model assumes that the error terms between $\varepsilon(\mathbf{x}^{(i)})$ and $\varepsilon(\mathbf{x}^{(j)})$ are correlated, and the correlation depends on the distance between these two points:

$$\mathrm{Corr}\left[\varepsilon(\mathbf{x}^{(i)}), \varepsilon(\mathbf{x}^{(j)})\right] = \exp\left(-\sum_{k=1}^{d} \theta_k |\mathbf{x}_k^{(i)} - \mathbf{x}_k^{(j)}|^{p_k}\right), \tag{2}$$

where d is the dimension of the design variables, θ_k and p_k $(k = 1, 2, \ldots, d)$ are parameters to be determined. Figure 1 shows the correlation function between $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ as $\theta = 1$ and $p = 2$. It is evident that the value of the correlation function is near one when the distance between the two points is small and approaches to zero when the distance becomes large. Moreover, the parameter θ_k measures the activity of the variable $\mathbf{x}_k$, and p_k represents the smoothness of the function in coordinate direction k. In fact, the correlation function is so powerful that it makes Kriging model very flexible to fit highly nonlinear and multimodal functions.

The $2d + 2$ parameters: $\mu, \sigma^2, \theta_1, \ldots, \theta_d, p_1, \ldots, p_d$ in the Kriging model are estimated by maximizing the likelihood of the sampled points. Then the prediction of an unknown point $\mathbf{x}$ as well as the mean squared error of the prediction are given by Kriging model (the detail derivation can be found in [21]):

$$\hat{y}(\mathbf{x}) = \hat{\mu} + \mathbf{r}^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{y} - \mathbf{1}\hat{\mu}) \tag{3}$$

and

$$s^2(\mathbf{x}) = \hat{\sigma}^2\left[1 - \mathbf{r}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{r} + \frac{\left(1 - \mathbf{1}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{r}\right)^2}{\mathbf{1}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{1}}\right]. \tag{4}$$

In the equations, $\mathbf{R}$ is a matrix with entry $R_{ij} = \mathrm{Corr}\left[\varepsilon\left(\mathbf{x}^{(i)}\right), \varepsilon\left(\mathbf{x}^{(j)}\right)\right]$, $\mathbf{r}$ is a n-dimensional vector with entry $r_i = \mathrm{Corr}\left[\varepsilon(\mathbf{x}), \varepsilon\left(\mathbf{x}^{(i)}\right)\right]$, $\mathbf{y} = \left(y^{(1)}, y^{(2)}, \ldots, y^{(n)}\right)$ is the vector of the n observed function values, and $\mathbf{1}$ is n-dimensional vector of ones.

2.2 Expected improvement criterion

The Kriging model provides not only the estimations about the prediction, but also the uncertainty of the prediction. The availability of the uncertainty makes Kriging model very suitable for designing efficient infill sampling criteria. Several infill criteria have been proposed based on the Kriging model, and the expected improvement function is the most covered criterion in literature.

Since the Kriging model treats the target function as a Gaussian process, the value of an unknown point $\mathbf{x}$ can be regarded as a random value $Y(\mathbf{x})$ of Gaussian distribution with mean $\hat{y}(\mathbf{x})$ and variance $s^2(\mathbf{x})$. Then the improvement of this point beyond the best observed value $f_{\min}$ is also a random value [5]:

$$I(\mathbf{x}) = \max\left(y_{\min} - Y(\mathbf{x}), 0\right). \tag{5}$$

The expected improvement criterion calculates the mathematic expectation of the improvement value, and can be derived in closed form [5]:

$$EI(\mathbf{x}) = \left(y_{\min} - \hat{y}(\mathbf{x})\right)\Phi\left(\frac{y_{\min} - \hat{y}(\mathbf{x})}{s(\mathbf{x})}\right) + s(\mathbf{x})\phi\left(\frac{y_{\min} - \hat{y}(\mathbf{x})}{s(\mathbf{x})}\right), \tag{6}$$

where $\Phi(\mathbf{x})$ and $\phi(\mathbf{x})$ are the cumulative density function and probability density function of the normal distribution respectively; $s(\mathbf{x})$ is the square root of the Kriging prediction variance.

The beauty of the expected improvement function is that it gives an elegant balance between local search and global search [5]. It can be seen in Eq. 6, the first term of the expected improvement criterion prefers points whose predictions $\hat{y}(\mathbf{x})$ are small, which will lead the search to local exploitation around the best observed point. The second term of the expected improvement criterion prefers points whose variances $s(\mathbf{x})$ are large, which will guide the search to global exploration at places where few points have been sampled before.

The expected improvement function of the Forrester function [16] are shown in Fig. 2. The function is $f(x) = (6x - 2)^2 \times \sin(2(6x - 2))$ [16] and the sampled points are $\mathbf{X} = [0, 0.5, 0.75, 1]^{\mathrm{T}}$. It can be seen that, the value of the expected improvement is zero at the sampled points, and rises up between them. The EI function of this example has three peaks, two of them are around the best observed point $x = 0.75$, and the other peak is located at the most thinly sampled area near the point $x = 0.25$. By preferring points around the best observed point and points far from the sampled points at the same time, the EI criterion gives a good trade-off between local search and global search.

2.3 Efficient global optimization algorithm

The efficient global optimization algorithm is a typical two-stage approach [5]. An initial Kriging model is built based on the initial sampled points in the first stage. In the second stage, one promising point is identified by optimizing the expected improvement function and is used to update the Kriging model in each cycle. The EGO algorithm proceeds in such

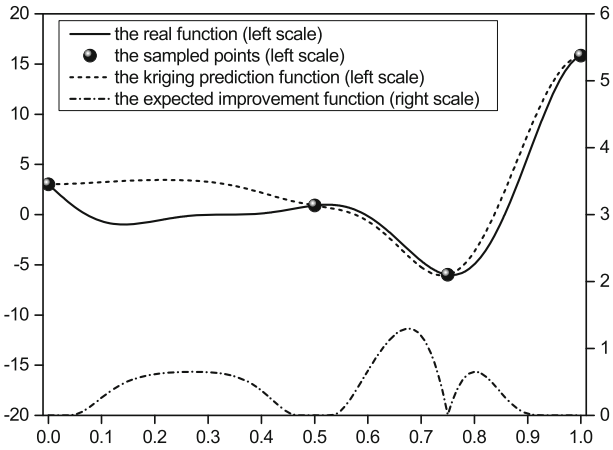

Fig. 2 The expected improvement function of the Forrester function. The function is $f(x) = (6x-2)^2 \times \sin(2(6x-2))$ [16], sampled points are $[0, 0.5, 0.75, 1]^{\mathrm{T}}$

a way to approach the global optimum of the original problem. The EGO algorithm [5] can be summed as Algorithm 1.

Algorithm 1 The framework of the EGO algorithm

Require: Initial design set $(\mathbf{X}, \mathbf{Y})$
Ensure: The best observation $(\mathbf{x}_{\min}, y_{\min})$
1: **while** the stop condition is not met **do**
2: Building a Kriging model based on the current design set $(\mathbf{X}, \mathbf{Y})$
3: $\mathbf{x}_{\text{new}} = \text{argmax } EI(\mathbf{x})$.
4: Evaluating $\mathbf{x}_{\text{new}}$ with the real function
5: $\mathbf{X} \leftarrow \mathbf{X} \cup \mathbf{x}_{\text{new}}$
6: $\mathbf{Y} \leftarrow \mathbf{Y} \cup y(\mathbf{x}_{\text{new}})$
7: $y_{\min} \leftarrow \min(\mathbf{Y})$
8: $\mathbf{x}_{\min} \leftarrow \mathbf{x} \in \mathbf{X} : y(\mathbf{x}) = y_{\min}$
9: **end while**

The value of the EI function is zero at sampled points and is positive at un-sampled places. By selecting the point with the highest EI value in each cycle, the updating points selected in the EGO process could be any point but the sampled points [5]. This means that the sampling of EGO algorithm is dense, and the convergence of EGO can be guaranteed according to the theory of Torn and Zilinskas [22].

One of the main limitations of the standard EI criterion is that it produces only one updating point in each cycle. As a result, the EGO algorithm can not evaluate the computationally expensive models in parallel in the second stage due to the point-to-point nature of the standard EI criterion. On the other hand, parallel computing architectures are commonly available and more reductions of the wall clock time are pursued in today's engineering industry. In order to make full use of the computing resources and further reduce the wall clock time, a new infill criterion named pseudo expected improvement (PEI) is proposed in this work to extend the traditional EGO algorithm to produce multiple points in each cycle. The PEI criterion is further explained in the following section.

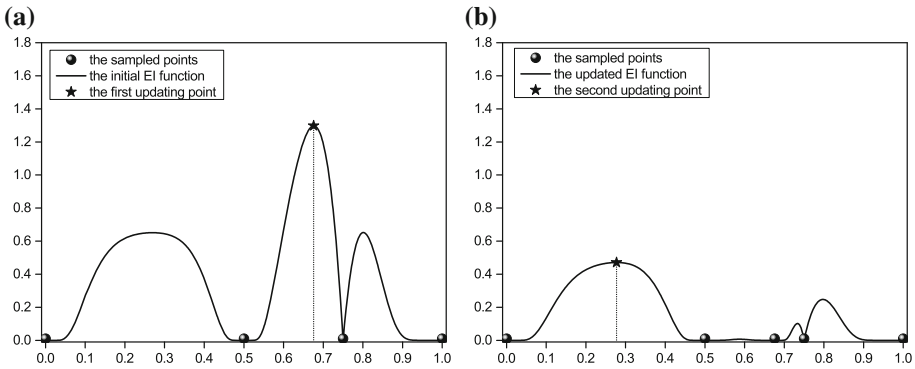

Fig. 3 **a** The initial expected improvement function; **b** the updated expected improvement function

3 The novel parallel efficient global optimization

3.1 The basic idea

The EGO algorithm chooses the point with the highest EI value to update the Kriging model and the EI function. After that the next candidate point can be selected based on the updated EI function. However, the traditional EGO algorithm can not get the second updating point without evaluating the first updating point because the EI function needs be updated by the first updating point. As a result, the traditional EGO algorithm can only evaluate the designs sequentially, not in a parallel way.

Figure 3a shows the initial EI function of the Forrester function [16]. The highest EI value is at the point $x = 0.676$. The EI function is updated, as shown in Fig. 3b, after the Kriging model is updated with the point $x = 0.676$. It is clear that when a point is used to update the Kriging model, the value of the EI function reduces drastically in the neighborhood of the updating point but changes slightly at places far away from the updating point. In other words, the influence that the updating point brings to the EI function at a certain point depends on the distance between this point and the updating point. The smaller the distance, the larger the influence.

The basic idea of this work is trying to approximate the updated EI function by tuning the initial EI function, more specifically, by multiplying it by an influence function. The influence function tries to simulate the impact that the updating point will bring to the EI function. By using the influence function, the approximated updated EI function can be calculated without evaluating the first updating point. Then the second updating point can be identified by maximizing the approximated updated EI function. In such a way, q updating points can be obtained within one cycle and evaluated simultaneously. The influence function could have any forms, but should have the following features.

1. The influence function should be continuous in the whole search space.
2. The value of the influence function at a point should be proportional to the distance between the point and the updating point. More specifically, the influence function should be zero at the updating point and be close to one at places far away from the updating point.
3. The influence function should be correlated only to the position of the updating point, not to the function value of the updating point.

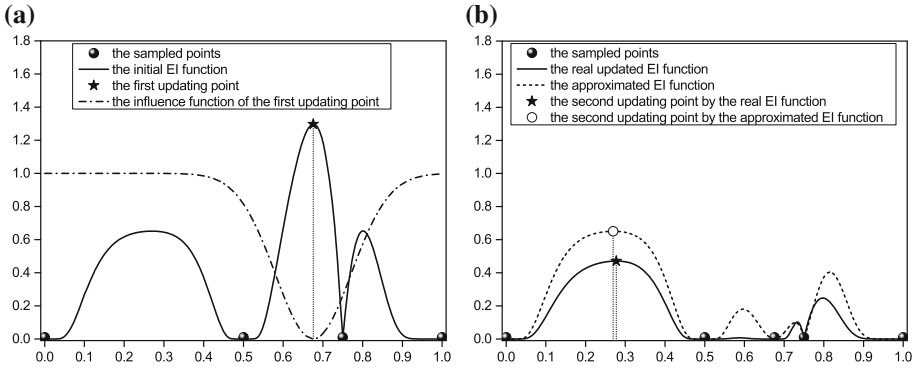

Fig. 4 **a** The initial expected improvement function; **b** the approximated expected improvement function

3.2 Influence function

An influence function which satisfies the above features is proposed in this work:

$$IF(\mathbf{x}, \mathbf{x}^u) = 1 - \mathrm{Corr}\left[\varepsilon(\mathbf{x}), \varepsilon(\mathbf{x}^u)\right], \tag{7}$$

where $\mathbf{x}^u$ is the updating point, $\mathrm{Corr}\left[\varepsilon(\mathbf{x}^{(i)}), \varepsilon(\mathbf{x}^{(j)})\right]$ is the correlation function between two points defined in Eq. 2. As described in Sect. 2.1, the correlation function is near one when the two points are really close to each other, and approaches to zero when the two points are very far from each other. As a result, the value of the influence function at the point $\mathbf{x}$ is zero if $\mathbf{x} = \mathbf{x}^u$, and approaches to one as $\|\mathbf{x} - \mathbf{x}^u\| \to \infty$. Besides, the influence function is a function of the updating point $\mathbf{x}^u$ and does not need any information about the function value of the updating point $y(\mathbf{x}^u)$. Obviously, the influence function is continuous in the whole design space. So the proposed influence function satisfies all the three features described in Sect. 3.1.

Figure 4a shows the initial EI function (the solid line) and the influence function (the dash dot line) of the first updating point $\mathbf{x}^u = 0.676$. Figure 4b shows the approximated updated EI function (the dash line) and the real updated EI function (the solid line). The approximated updated EI function is calculated by multiplying the initial EI function by the influence function of the first updating point $\mathbf{x}^u = 0.676$. The approximated updated EI function is not exactly the same as the real updated EI function, but the trends of the two functions are similar. More importantly, they suggest approximately the same candidate point in next cycle.

It is reasonable that there is some difference between the approximated updated EI function and the real updated EI function, because the real influence that the updating point will have on the EI function can not be known before evaluating the updating point. The major purpose of this work is trying to produce multiple promising candidates at each iteration, not to build a highly accurate EI function. Therefore, the difference between these two functions is not a problem from this point of view.

3.3 Pseudo expected improvement criterion

Based on the influence function, a novel infill criterion for sequentially selecting q candidate points is developed. Assume that n initial design points $\left\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n)}\right\}$ are used to build the Kriging model. The initial EI function can be calculated as in Eq. 6, and the first

updating point is identified by maximizing the initial EI function just like the traditional EGO algorithm does:

$$\mathbf{x}^{(n+1)} = \text{argmax } EI(\mathbf{x}). \tag{8}$$

Then instead of evaluating the point $\mathbf{x}^{(n+1)}$ to update the EI function, an approximation of the updated EI function is calculated by multiplying the initial EI function by the influence function of the point $\mathbf{x}^{(n+1)}$:

$$PEI\left(\mathbf{x}, \mathbf{x}^{(n+1)}\right) = EI(\mathbf{x}) \cdot IF\left(\mathbf{x}, \mathbf{x}^{(n+1)}\right). \tag{9}$$

The approximated updated EI function is called pseudo expected improvement (PEI) function in this work because it is not a 'real' expected improvement function. The second updating point is produced by maximizing the approximated updated EI function:

$$\mathbf{x}^{(n+2)} = \text{argmax } PEI\left(\mathbf{x}, \mathbf{x}^{(n+1)}\right). \tag{10}$$

As the process goes on, the PEI function can be derived after $q-1$ points have been added to the design set:

$$\begin{aligned} PEI\left(\mathbf{x}, \mathbf{x}^{(n+1)}, \mathbf{x}^{(n+2)}, \ldots, \mathbf{x}^{(n+q-1)}\right) = EI(\mathbf{x}) \cdot IF\left(\mathbf{x}, \mathbf{x}^{(n+1)}\right) \\ \cdot IF\left(\mathbf{x}, \mathbf{x}^{(n+2)}\right) \cdot IF\left(\mathbf{x}, \mathbf{x}^{(n+q-1)}\right), \end{aligned} \tag{11}$$

and the qth updating point is identified as:

$$\mathbf{x}^{(n+q)} = \text{argmax } PEI\left(\mathbf{x}, \mathbf{x}^{(n+1)}, \mathbf{x}^{(n+2)}, \ldots, \mathbf{x}^{(n+q-1)}\right). \tag{12}$$

For simplify, the PEI function is defined as:

$$PEI(\mathbf{x}, q-1) = EI(\mathbf{x}) \cdot IF\left(\mathbf{x}, \mathbf{x}^{(n+1)}\right) \cdot IF\left(\mathbf{x}, \mathbf{x}^{(n+2)}\right) \cdots IF\left(\mathbf{x}, \mathbf{x}^{(n+q-1)}\right), \tag{13}$$

where the second parameter $q-1$ of the PEI function indicates the number of updating points that have been added to the model. Plug the influence function in Eq. 7 and the correlation function in Eq. 2 into this formula, we can get:

$$\begin{aligned} PEI(\mathbf{x}, q-1) &= EI(\mathbf{x}) \cdot IF\left(\mathbf{x}, \mathbf{x}^{(n+1)}\right) \cdot IF\left(\mathbf{x}, \mathbf{x}^{(n+2)}\right) \cdots IF\left(\mathbf{x}, \mathbf{x}^{(n+q-1)}\right) \\ &= EI(\mathbf{x}) \cdot \prod_{i=1}^{q-1} \left[1 - \text{Corr}\left[\varepsilon(\mathbf{x}), \varepsilon\left(\mathbf{x}^{(n+i)}\right)\right]\right] \\ &= EI(\mathbf{x}) \cdot \prod_{i=1}^{q-1} \left[1 - \exp\left(-\sum_{k=1}^{d} \theta_k \left|\mathbf{x}_k - \mathbf{x}_k^{(n+i)}\right|^{p_k}\right)\right], \end{aligned} \tag{14}$$

where θ_k and p_k $(k = 1, 2, \ldots, d)$ are parameters defined in Eq. 2. The only parameters θ_k and p_k used in the PEI function are determined after the Kriging model is built. Therefore, no parameter is needed to be specified by the user when using the PEI criterion, which makes the PEI criterion very easy to implement.

When the number of updating points q is set to 0, indicating no updating point is added to the model, the PEI function degenerates to the standard EI function $PEI(\mathbf{x}, 0) = EI(\mathbf{x})$. In other words, the standard EI criterion is a special case of the proposed PEI criterion.

3.4 Proposed parallel EGO algorithm

A parallel EGO algorithm called EGO–PEI algorithm is proposed based on the PEI criterion. The EGO–PEI approach is summed up in Algorithm 2.

Algorithm 2 The framework of the novel proposed parallel EGO algorithm

Require: Initial design set $(\mathbf{X}, \mathbf{Y})$; number of updating points q
Ensure: The best observation $(\mathbf{x}_{\min}, y_{\min})$
while the stop condition is not met **do**
  Building a Kriging model based on the current design set $(\mathbf{X}, \mathbf{Y})$
  for $i = 1$ to q **do**
    $\mathbf{x}^{(n+i)} = \text{argmax } PEI(\mathbf{x}, i-1)$.
  end for
  Evaluating $\{\mathbf{x}^{(n+1)}, \ldots, \mathbf{x}^{(n+q)}\}$ in parallel with the real function
  $\mathbf{X} \leftarrow \mathbf{X} \cup \{\mathbf{x}^{(n+1)}, \ldots, \mathbf{x}^{(n+q)}\}$
  $\mathbf{Y} \leftarrow \mathbf{Y} \cup \{y(\mathbf{x}^{(n+1)}), \ldots, y(\mathbf{x}^{(n+q)})\}$
  $y_{\min} \leftarrow \min(\mathbf{Y})$
  $\mathbf{x}_{\min} \leftarrow \mathbf{x} \in \mathbf{X} : y(\mathbf{x}) = y_{\min}$
end while

The PEI function should be optimized q times in order to produce q candidate points by the EGO–PEI approach in each cycle. Fortunately, the computation of PEI function is very cheap, so the optimization time of PEI function can be neglected compared to the time of evaluating the computationally expensive models. Since the value of the PEI function at the updated points is zero (the first term of Eq. 14 is zero at points $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n)}\}$ and the second term is zero at points $\{\mathbf{x}^{(n+1)}, \mathbf{x}^{(n+2)}, \ldots, \mathbf{x}^{(n+q-1)}\}$), the next sampling point selected by the PEI criterion would be anywhere but the sampled points. Therefore, the sampling of the proposed EGO–PEI algorithm is dense and the convergence of this algorithm can be guaranteed according to the theory of [22].

3.5 Relationship to another sequential approach

Another sequential approach which is similar to the proposed EGO–PEI approach is the *Kriging Believer* or the *Constant Liar* approach proposed by Ginsbourger [13]. Both approaches select q candidate points sequentially in each cycle, and both approaches need q optimizations to get q candidate points. The difference between these two algorithms is also significant. The *Kriging Believer* or the *Constant Liar* approach updates the Kriging model with a 'fake' point to calculate the next EI function. While in the proposed approach, the next EI function is approximated by simply multiplying the initial EI function by the influence function. It means that the hyper-parameters of the Kriging model need to be calculated $q - 1$ times in Ginsbourger's approach, but need not in the proposed approach. The calculation of the hyper-parameters often involves an optimization work which may cost time when the dimension of the problem is high.

Ginsbourger [13] compared the *Kriging Believer* algorithm and three *Constant Liar* algorithms with settings of $L = \max(\mathbf{Y})$, $L = \text{mean}(\mathbf{Y})$ and $L = \min(\mathbf{Y})$ respectively. The results show that the *Constant Liar* algorithm with setting of $L = \min(\mathbf{Y})$ gives the best results. Therefore, only the *Constant Liar* algorithm with setting of $L = \min(\mathbf{Y})$ (it is called CL[min] algorithm for short in this paper) is included in the study. In order to study the efficiency of the proposed EGO–PEI algorithm, we compare it against the standard EGO algorithm as well as the CL[min] algorithm through numerical experiments.

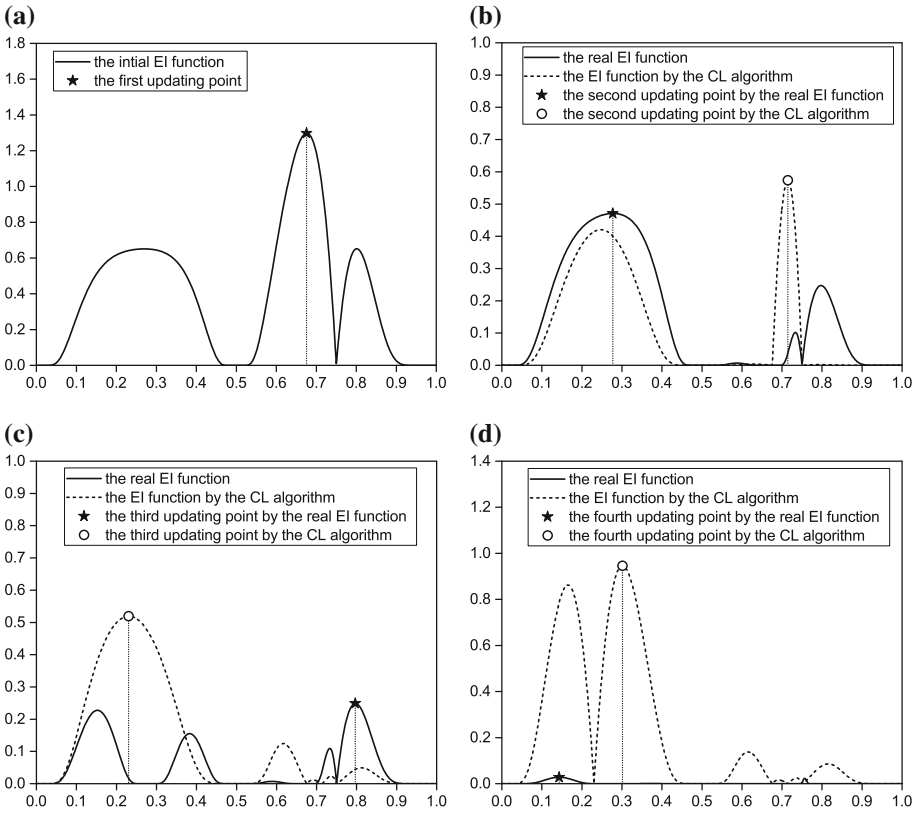

Fig. 5 The CL[min] algorithm on Forrester function [16] compared with the standard EGO algorithm. **a** The first iteration; **b** the second iteration; **c** the third iteration; **d** the fourth iteration

3.6 Demonstrative example

First, we apply the CL[min] algorithm and the proposed EGO–PEI algorithm on the one-dimensional Forrester function [16]. The initial sampled points are $\mathbf{X} = [0, 0.5, 0.75, 1]^{\mathrm{T}}$. The four candidate points produced in the first cycle by the CL[min] algorithm and the EGO–PEI algorithm are shown in Figs. 5 and 6 respectively. Besides, the four updating points selected by the standard EGO algorithm are depicted in the figures as a reference.

Figure 5 shows the CL[min] algorithm on the Forrester function [16]. The first candidate point is selected by the standard EI criterion as shown in Fig. 5a. After that, the CL[min] algorithm updates the Kriging model with the 'fake' point $(x, y) = (0.676, y_{\min})$, and obtains the second EI function (the dash line in Fig. 5b). It is clear in Fig. 5b that, the second EI function by the CL[min] algorithm is quite different from the real EI function. After using the 'fake' point $x = 0.676$ with value of $y_{\min}$, the CL[min] algorithm thinks there is more improvement near the point $x = 0.676$, where there is actually few improvement. At the third iteration, the CL[min] algorithm turns to the sparsely sampled region around the point $x = 0.25$. However, after updating the Kriging model with the liar, two higher peaks appear in the EI function around the point $x = 0.25$ (see Fig. 5d). Using the $y_{\min}$ value to update the

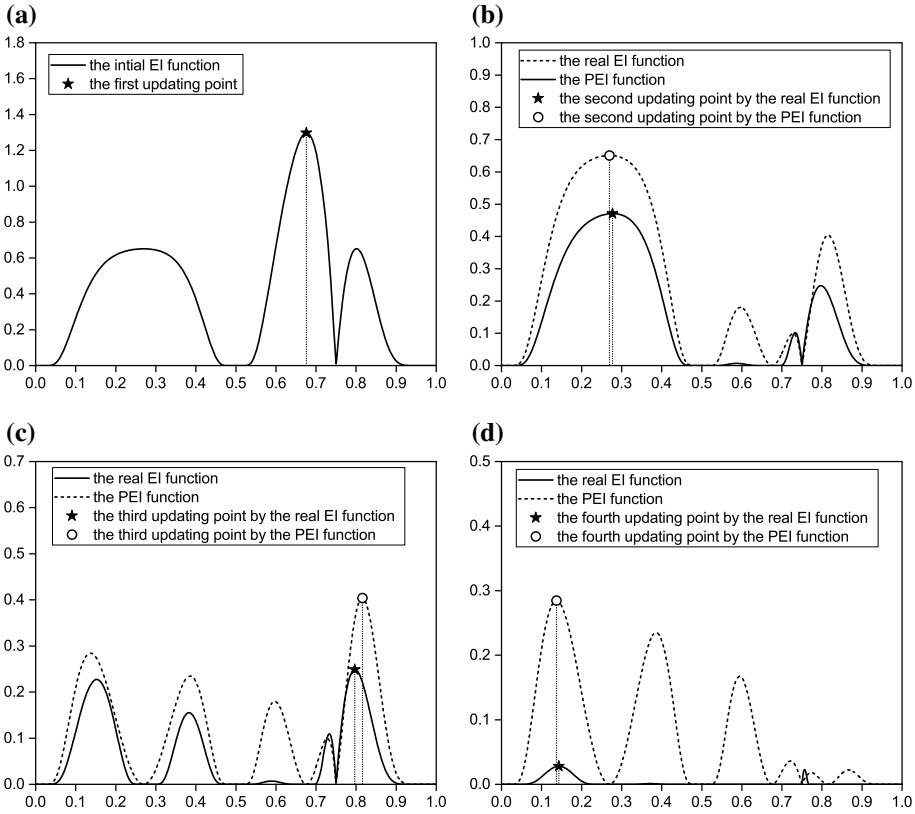

Fig. 6 The EGO–PEI algorithm on Forrester function [16] compared with the standard EGO algorithm. **a** The first iteration; **b** the second iteration; **c** the third iteration; **d** the fourth iteration

Kriging model can easily fool the search to believe that more improvement can be achieved around the 'fake' minimum, which makes the CL[min] algorithm very obsessed to local search.

Figure 6 illustrates the search process of the EGO–PEI algorithm on the Forrester function [16]. The trends of the PEI functions are similar to the real EI functions, and the four candidate points selected by the PEI criterion are really close to the points selected by the standard EI criterion.

Normally, in the standard EGO algorithm, as the updating process goes on, the value of the EI function decreases gradually. Since the value of an influence function is between zero and one, as the selecting process goes on, more influence functions will be multiplied to the initial EI function and the value of PEI function will decrease gradually (see Fig. 6). However, the EI functions of the CL[min] algorithm do not decrease monotonously as can be seen in Fig. 5. The value of the EI function sometimes increases after adding the 'fake' points at some region. Another issue with the *Constant Liar* approach is that the lie value needs to be set properly so that the *Constant Liar* approach can work efficiently. The theoretically best setting is problem-dependent and can not be known in advance. On the contrary, the PEI criterion has no user-defined parameters, thus is easier to implement.

4 Numerical experiments

4.1 Test problems

Six analytic problems are used for the experiments so that we do not need to worry the computational time. The six test problems are described below:

1. Six-hump camel-back function (Sixhump) with $n = 2$ [23]:

$$f = 4x_1^2 - \frac{21}{10}x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$$
$$x_1, x_2 \in [-2, 2]. \tag{15}$$

2. Branin function with $n = 2$ [23]:

$$f = \left(x_2 - \frac{5}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10$$
$$x_1 \in [-5, 10], x_2 \in [0, 15]. \tag{16}$$

3. Sasena function with $n = 2$ [24]:

$$f = 2 + 0.01(x_2 - x_1^2)^2 + (1 - x_1)^2 + 2(2 - x_2)^2 + 7\sin(0.5x_1)\sin(0.7x_1x_2)$$
$$x_1, x_2 \in [0, 5]. \tag{17}$$

4. Goldstein-Price function (GoldPrice) with $n = 2$ [23]:

$$f = \left[1 + (x_1 + x_2 + 1)^2\left(19 - 14x_1 + 3x_1^2 - 14x_2 + 16x_1x_2 + 3x_2^2\right)\right]$$
$$\times\left[30 + (2x_1 - 3x_2)^2\left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right)\right]$$
$$x_1, x_2 \in [0, 5]. \tag{18}$$

5. Hartman function with $n = 3$ (Harman3) and $n = 6$ (Hartman6) [23]:

$$f = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{n}\alpha_{ij}(x_j - p_{ij})^2\right]$$
$$x_i \in [0, 1], i = 1, 2, \ldots, n. \tag{19}$$

where c_i are elements of vector $\mathbf{c}$; α_{ij} and p_{ij} are the elements of matrices $\mathbf{A}$ and $\mathbf{P}$, respectively. For both Hartman functions $\mathbf{c} = [1.0, 1.2, 3.0, 3.2]$. $\mathbf{A}$ and $\mathbf{P}$ are shown in Table 1.

4.2 Building the Kriging model

For all experiments, $10d$ (d is the dimension of the test problems) initial designs are created to build the initial Kriging model. The MATLAB® Latin hypercube design function *lhsdesign* is used to produce the initial designs with *maximin* option with 1000 iterations. In the experiments, the Kriging models are built using the DACE (Design and Analysis of Computer Experiments) toolbox [25] with the following setting:

- Regression function: *regpoly0*.
- Correlation function: *corrgauss*.
- Initial guess on θ: $(10d)^{-1/d}$

Table 1 Parameters of Hartman3 and Hartman6

Functions	Parameters
Hartman3	$\mathbf{A} = \begin{bmatrix} 3.0 & 10.0 & 30.0 \\ 0.1 & 10.0 & 35.0 \\ 3.0 & 10.0 & 30.0 \\ 0.1 & 10.0 & 35.0 \end{bmatrix}$
	$\mathbf{P} = \begin{bmatrix} 0.3689 & 0.1170 & 0.2673 \\ 0.4699 & 0.4378 & 0.7470 \\ 0.1091 & 0.8732 & 0.5547 \\ 0.03815 & 0.5743 & 0.8828 \end{bmatrix}$
Hartman6	$\mathbf{A} = \begin{bmatrix} 10.0 & 3.0 & 17.0 & 3.5 & 1.7 & 8.0 \\ 0.05 & 10.0 & 17.0 & 0.1 & 8.0 & 14.0 \\ 3.0 & 3.5 & 1.7 & 10.0 & 17.0 & 8.0 \\ 17.0 & 8.0 & 0.05 & 10.0 & 0.1 & 14.0 \end{bmatrix}$
	$\mathbf{P} = \begin{bmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5886 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1451 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{bmatrix}$

4.3 Optimizing the expected improvement function

The expected improvement function as well as the pseudo expected improvement function are optimized by the differential evolution (DE) algorithm [26,27]. The DE algorithm is executed with the following settings:

- Population size: 50.
- Maximum iterations: 100.
- DE-stepsize: 0.8.
- Crossover probability: 0.8.
- Strategy: DE/rand/1/bin.

Since multiple independent optimizations can increase the chance of success in finding the global optimum, the DE algorithm is run four times to take the best solution.

4.4 Running EGO and parallel EGO algorithms

Both the two parallel EGO algorithms are run to produce 2, 4, 6, 8 and 10 points in each cycle. For all compared algorithms, another 400 evaluations are allowed for all test problems, which means maximum 400 optimization cycles for the standard EGO algorithm and 40 optimization cycles for 10-point parallel EGO algorithms. To average out the influence of the initial designs, the experiments are repeated 100 times with 100 different initial designs.

4.5 Comparison metric

The compared metric is the number of cycles an algorithm needs to find the optimum of the test functions within 1%. The maximum number of cycles is counted if a run fails to find the optimum. Note that, the number of evaluations is equal to q (q is the number of updating points used in each cycle) times of the number of cycles. Since it is easy to get the number of evaluations from the number of cycles, only the number of cycles is displayed in this paper.

Table 2 The number of cycles (median, mean and standard deviation of 100 runs) needed to find the real optimum of the test problems within 1%

Test problems	Number of cycles	EGO	EGO–PEI				
			2-point	4-point	6-point	8-point	10-point
Sixhump	Median	9	5	3	3	2	2
	Mean	8.05	4.43	2.90	2.48	2.25	2.00
	SD	2.15	1.27	0.85	0.67	0.67	0.55
Branin	Median	26	14	7	6	5	4
	Mean	25.75	13.51	7.34	5.66	4.69	4.12
	SD	4.51	2.23	1.36	1.01	0.76	0.59
Sasena	Median	29	16	9	7	6	5
	Mean	30.22	15.95	9.22	6.82	5.95	5.31
	SD	9.96	5.34	2.59	2.22	1.67	1.59
GoldPrice	Median	55.5	29	15	11	9	7
	Mean	60.42	30.38	15.51	11.28	9.11	7.68
	SD	21.24	8.50	3.33	2.15	1.73	1.07
Hartman3	Median	17	12	7	5	4	4
	Mean	20.92	12.04	7.00	5.31	4.48	4.02
	SD	15.78	7.98	3.85	2.55	1.90	1.66
Hartman6	Median	18	10	7	6	5	5
	Mean	56.09	31.14	19.48	12.46	10.69	8.96
	SD	77.34	39.17	25.09	15.59	13.11	9.89

5 Results and discussions

5.1 Comparison with traditional EGO algorithm

The proposed EGO–PEI algorithm is first compared to the standard EGO algorithm. The compared metric is the number of cycles that the two algorithms need to find the optimum of a test problem. The median, mean and standard deviation of 100 runs are shown in Table 2.

For all the six test problems, the proposed EGO–PEI algorithm performs much better than the traditional EGO algorithm no matter how many updating points are selected in each cycle. Significant improvements can be observed. For example, the standard EGO algorithm needs an average 8.05 cycles to find the optimum of the Sixhump function, while the 10-point EGO algorithm needs only 2 cycles averagely. For GoldPrice function, the data is 60.42–7.68, and is 56.09–8.96 for Hartman6 function. The results indicate that the proposed influence function is capable of simulating the impact that the updating points bring to the expected improvement function, and the proposed PEI criterion is efficient to generate multiple candidate points on the six test problems. As the number of updating points increases, all the three statistics (the median, mean and standard deviation) of the number of cycles reduce continuously. In other words, the proposed EGO–PEI algorithm becomes more efficient (the average number of cycles goes down) and robust (the standard deviation of number of cycles goes down) on the test functions when more points are added in each cycle.

However, compared to the standard EGO algorithm, the reduction of cycles by the proposed EGO–PEI algorithm is not strictly in proportion to the number of points used in

each cycle. For example, averagely, the 2-point EGO–PEI algorithm speeds up the search $8.05/4.45 \approx 1.81$ times on Sixhump function, which should be 2 times for a *linear speedup* algorithm [28,29]. The 10-point EGO–PEI speeds up the search $56.09/8.96 \approx 6.26$ times on the Hartman6 problem, which should be 10 times for an ideal parallel algorithm. As the number of updating points increases, the performance of the proposed EGO–PEI algorithm improves gradually. However, the improvement becomes slow when the number of updating points becomes high. The reason may be that as the sequential selecting process goes on within one cycle, more and more influence functions are multiplied to the initial EI function, which makes the PEI function more and more unreliable. As a result, the candidate points selected by the PEI criterion become more and more unpromising. Therefore, the average efficiency of the updating points goes down and the improvement of the proposed EGO–PEI algorithm becomes slow.

Since the speedup of the proposed EGO–PEI algorithm is below *linear speedup* [28,29], the number of evaluations that the EGO–PEI algorithm needs to find the optimum would be greater than the number of evaluations that the standard EGO algorithm needs. However, in today's engineering industry, where parallel computing architectures are often available, the dominant cost is the wall clock time (the number of cycles) rather than the computational resources (the number of evaluations). The proposed EGO–PEI algorithm is able to accelerate the convergence and save the wall clock time compared to the traditional EGO algorithm, therefore, is more suitable for engineering design when parallel computing is available.

5.2 Comparison with the constant liar algorithm

Then the proposed EGO–PEI algorithm is compared to the CL[min] algorithm. The compared metric is the number of cycles that the compared algorithms need to find the optima of the test functions. The median, mean and standard deviation of 100 runs are shown in Table 3. In addition, the paired t-tests are conducted to identify whether or not the results of the two compared algorithms are significantly different. The null hypothesis of the paired t-test is that the results of the proposed EGO–PEI algorithm and the CL[min] algorithm have the same means. If the calculated p-value is smaller than the significance level $\alpha = 0.05$, then the null hypothesis is rejected, which implies that the two algorithms gain significantly different results. The p-values of the significant tests are given in Table 3, and the significantly smaller means are bold when the p-values are smaller than $\alpha = 0.05$.

It can be seen that the proposed EGO–PEI algorithm performs similarly to the CL[min] algorithm on the Sixhump function, and performs significantly better than the CL[min] algorithm in most cases on the other five test problems at the significance level of $\alpha = 0.05$. There is no case that the CL[min] algorithm outperforms the proposed EGO–PEI algorithm significantly on all the 6 test problems regardless of how many updating points are used per cycle.

The proposed EGO–PEI algorithm seems to have more advantages than the CL[min] algorithm on hard and high-dimensional problems. The Sixhump problem is easy to solve, both algorithms can reach the optimum within averagely 2 cycles using 10 updating points in each cycle. The paired t-tests show that there is no significant difference between the results of the compared algorithms on the Sixhump function no matter how many updating points are used in each cycle. The 2-dimensional Branin, Sasena and GoldPrice problems are thought to be harder to solve, and the EGO–PEI algorithm outperforms the CL[min] algorithm on these problems in at least 4 of the 5 cases. For the test instances having dimension higher than two, the improvements of the EGO–PEI algorithm beyond the CL[min] algorithm become clearer. The EGO–PEI algorithm needs significantly fewer number of cycles than the CL[min]

Table 3 The number of cycles (median, mean and standard deviation of 100 runs) needed to find the real optimum of the test problems within 1%, and the p-values of the paired t-tests

Test problems	Number of cycles	2-point		4-point		6-point		8-point		10-point	
		CL	PEI	CL	PEI	CL	PEI	CL	PEI	CL	PEI
Sixhump	Median	5	5	3	3	3	3	2	2	2	2
	Mean	4.62	4.43	2.96	2.90	2.42	2.48	2.18	2.25	1.94	2.00
	SD	1.45	1.27	0.95	0.85	0.87	0.67	0.78	0.67	0.63	0.55
	p-value	0.1880		0.5883		0.5551		0.4889		0.4254	
Branin	Median	14.5	14	8	7	6	6	5	5	4	4
	Mean	14.23	**13.51**	7.90	**7.34**	5.82	5.60	4.96	**4.69**	4.39	**4.12**
	SD	2.22	2.23	1.50	1.36	0.95	1.01	1.10	0.76	0.89	0.59
	p value	0.0071		0.0037		0.2046		0.0454		0.0084	
Sasena	Median	17	16	10	9	8	7	6	6	6	5
	Mean	17.34	**15.95**	10.58	**9.22**	7.95	**6.82**	6.51	**5.95**	5.73	**5.31**
	SD	5.21	5.34	3.39	2.60	2.71	2.23	2.22	1.67	1.54	1.59
	p-value	0.0066		0.0001		0.0001		0.0142		0.0265	
GoldPrice	Median	28.5	29	16	15	11	11	9	9	8	7
	Mean	31.70	30.38	17.99	**15.54**	12.20	**11.28**	10.24	**9.11**	8.93	**7.68**
	SD	11.64	8.50	6.69	3.33	3.93	2.15	3.80	1.73	3.43	1.07
	p-value	0.3579		0.0010		0.0338		0.0080		0.0007	
Hartman3	Median	13	12	9	7	6.5	5	5	4	6	4
	Mean	13.46	12.04	9.45	**7.00**	7.44	**5.31**	5.75	**4.48**	5.31	**4.02**
	SD	8.85	7.98	6.07	3.85	4.94	2.55	3.37	1.90	3.09	1.66
	p-value	0.0768		0.0000		0.0000		0.0003		0.0000	
Hartman6	Median	11	10	7	7	6	6	5	5	5	5
	Mean	36.74	31.14	27.57	**19.48**	21.58	**12.46**	14.00	10.69	13.81	**8.96**
	SD	52.59	39.17	33.61	25.09	25.18	15.59	17.34	13.11	14.75	9.89
	p-value	0.1013		0.0005		0.0000		0.0769		0.0005	

algorithm to converge on the Hartman3 and Hartman6 problems in most cases. The average number of cycles is 4.02–5.31 on the Hartman3 function and is 8.96–13.81 on the Hartman6 function when 10 updating points are used in each cycle.

Another phenomenon is that the EGO–PEI algorithm seems to be more efficient than the CL[min] algorithm when the number of updating points is high. When 2 updating points are used in each cycle, the EGO–PEI algorithm outperforms the CL[min] algorithm significantly on 2 of the 6 test problems at the significance level of $\alpha = 0.05$. By comparison, the EGO–PEI algorithm outperforms the CL[min] algorithm at least on 4 of the 6 test instances when 4 or more updating points are used in each cycle. This indicates that the PEI criterion is more efficient in producing large number of updating points than the CL[min] algorithm.

5.3 Discussion about the optimizer of the PEI criterion

In theory, the q (q is the number of updating points used in each cycle) candidate points suggested by the PEI criterion are determined once the initial EI function is built, if the global solutions of all the PEI functions (including the initial EI function) can be found. However, in practice, the PEI function is often highly multimodal and its global optimum can not be guaranteed to be found especially when the dimension of the problem is high. If the global solution of the first PEI function (the initial EI function) is not found, it will affect the calculation of the second PEI function. Then, the second PEI function will produce a biased candidate point, and affect the third PEI function. As the process goes on, the bias of the first updating point will propagate to the calculation of the final PEI function.

This phenomenon is shown on the one-dimensional Forrester function [16] in Fig. 7. The real global maximum of the initial EI function is at the point $x = 0.676$, but the obtained solution is at the point $x = 0.596$. As the sequential process goes on, the bias of the first updating point affects the calculation of the following PEI functions. It can be seen in Fig. 7 that the biased PEI functions in the second iteration to the fourth iteration are quite different from the real PEI functions. However, the biased PEI functions suggest very similar candidate points as the real PEI functions do. This example shows that the PEI function has some internal robustness even if the global maximum of the initial EI function is not found.

Unfortunately, this effect can not be analyzed analytically since it highly depends on the properties of the initial EI function and the bias of the first updating point. However, there is another way to see this problem. Since the value of the PEI function (the EI function) indicates how much improvement the candidate point can bring to the search, when the global maximum of a PEI function is not found, the effect on the convergence can be interpreted as the decrease of the average quality of the candidate points. From this point of view, the effect on the convergence can be studied by comparing the same optimizer with different qualities. In this work, we run the EGO–PEI algorithm on the six benchmarks using the DE algorithm with five different settings: $10d \times 400$ (population size $\times$ maximum iterations), $10d \times 200$, $10d \times 100$, $10d \times 50$ and $10d \times 25$, where d is the dimension of the test problems. 10 updating points are selected in each cycle in the experiments.

Table 4 shows the statistical data of the number of cycles that the EGO–PEI algorithm needs to converge on the six benchmarks using the DE algorithm with the five different settings. In addition, the one-way ANOVA (analysis of variance) test is used to identify whether or not the results of the five groups (each group with a different setting) are significantly different. The null hypothesis of the ANOVA test is that the results of the 5 different groups have the same means. Once the calculated p-value is smaller than the significance level of $\alpha = 0.05$, we reject the null hypothesis and interpret that at least the result of one group is significantly different from at least one other group.

Table 4 The number of cycles (median, mean and standard deviation of 100 runs) needed to find the real optimum of the test problems within 1%, and the p-values of the one-way ANOVA tests

Test problems	Number of cycles	Optimizer with different settings				
		$10d \times 400$	$10d \times 200$	$10d \times 100$	$10d \times 50$	$10d \times 25$
Sixhump	Median	2	2	2	2	2
	Mean	2.03	2.03	2.06	2.12	2.34
	SD	0.6106	0.5938	0.6000	0.5908	0.9663
	p-value	0.0067				
Branin	Median	4	4.5	4	5	7
	Mean	4.47	4.58	4.51	4.78	8.71
	SD	0.9370	0.9121	0.8102	1.0691	5.2499
	p-value	0.0000				
Sasena	Median	5.5	6	5	6	6
	Mean	5.51	5.53	5.57	5.59	6.21
	SD	1.4736	1.4527	1.7306	1.6149	1.7826
	p-value	0.0097				
GoldPrice	Median	8	8	8	8	12
	Mean	8.10	8.11	8.00	8.60	16.93
	SD	1.4178	1.4764	1.4907	2.4329	10.5018
	p-value	0.0000				
Hartman3	Median	4	4	4	4	4
	Mean	4.22	4.21	4.16	4.23	4.48
	SD	1.7842	1.7191	1.7963	1.7457	1.9974
	p-value	0.7481				
Hartman6	Median	5	5	5	40	40
	Mean	6.15	6.27	12.17	35.98	40.00
	SD	4.9836	4.9805	13.8819	11.0298	0.0000
	p-value	0.0000				

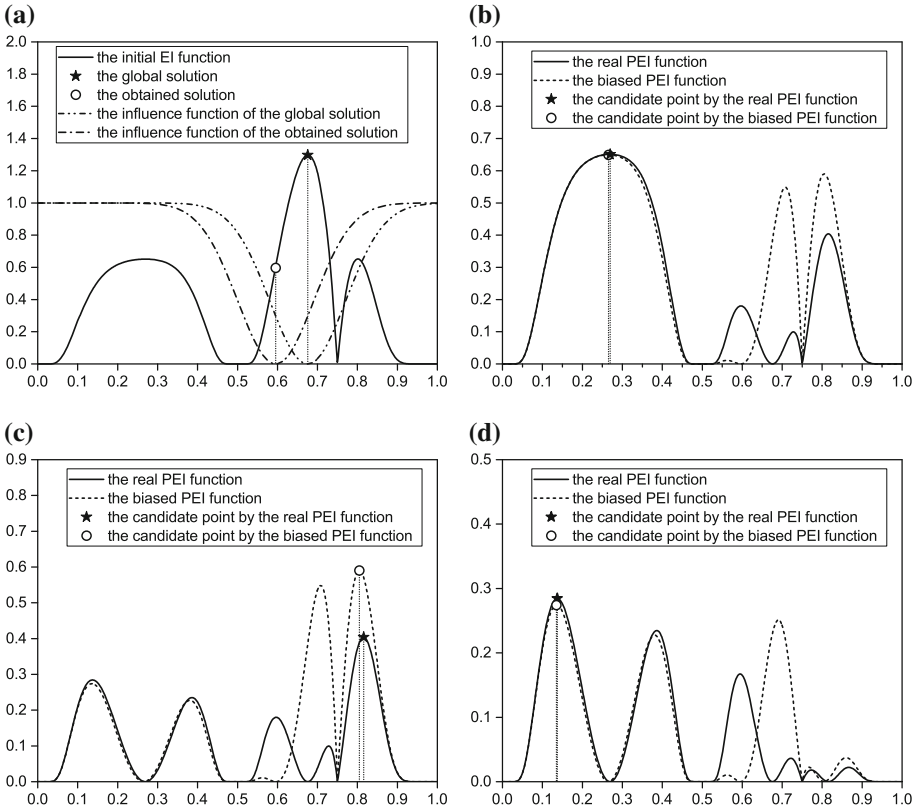

Fig. 7 The PEI functions on the Forrester function [16] when the optimum of the initial EI function is not found. **a** The first iteration; **b** the second iteration; **c** the third iteration; **d** the fourth iteration

Overall, it can be seen that the higher the quality of the optimizer, the faster the convergence of the algorithm. For the four 2-dimensional benchmarks, the p-values are all smaller than the significance level of $\alpha = 0.05$, which means that the quality of the optimizer has a significant influence on the convergence of the EGO–PEI algorithm. It is interesting to see that the results among the first four groups are very similar, but dramatically worsen in the fifth group where the DE algorithm with setting $10d \times 25$ is used. The calculated p-values of the first four groups are 0.6812, 0.0920, 0.9835 and 0.0713, for the Sixhump, Branin, Sasena and GoldPrice problems respectively. This indicates that the effect on the convergence is not significant when the iteration number of the DE algorithm changes from 400 to 50, but the effect is significant when the iteration number changes from 50 to 25. The reason may be that the DE algorithm can not converge to the global maximum of the PEI function within 25 iterations, but can converge with 50 iterations. Therefore, increasing the number of iterations higher than 50 can not further improve the performance of the EGO–PEI algorithm. Although using the DE algorithm with 25 iterations needs significant more cycles to find the global solutions of the four 2-dimensional problems, it can still converge to the global optima of the test problems.

The p-value is higher than the significance level of $\alpha = 0.05$ on the Hartman3 function, which indicates that quality of the optimizer has no significant effect on the convergence of the EGO–PEI algorithm over the Hartman3 problem. The reason might be that the PEI function of the Hartman3 function is easy to optimize and the DE algorithm can converge to the maximum even within 25 iterations. For the Hartman6 problem, the influence of the quality

of the optimizer is very significant. Using the DE algorithm with 400 and 200 iterations gains very similar and good results, but significantly worse results with 100 iterations. When the iteration number goes down below 50, the EGO–PEI algorithm can hardly converge to the optimum of the Hartman6 function.

In summary, the quality of the optimizer really has a significant effect on the performance of the EGO–PEI algorithm, especially on the Hartman6 problem. Since the PEI function is cheap to calculate, it is recommended to use as much evaluations as one can afford to find the global maximum of the PEI function. Although using more evaluations of the PEI function will increase the optimization time, it can reduce the number of evaluations of the computationally expensive models, which will gain a large amount of reduction of time in return.

6 Conclusions

In this work, a novel infill criterion called pseudo expected improvement (PEI) criterion is proposed for developing EGO algorithms. In each cycle, the first updating point is selected by the standard EI function, then the PEI function is built to approximate the real updated EI function by multiplying the initial EI function by the influence function of the first updating point. The influence function is designed to simulate the impact that the updating point will have on the EI function, and is only corresponding to the position of the updating point. Therefore, the next updating point can be identified by the PEI function without evaluating the first updating point. As the process goes on, a desired number of updating points can be selected by the PEI function.

The efficiency of the EGO–PEI algorithm is validated over six test problems with dimension from 2 to 6. First, the EGO–PEI algorithm is compared against the standard EGO algorithm. The results show that the proposed EGO–PEI algorithm is more efficient and robust than the standard EGO algorithm on all the six test problems. Then the EGO–PEI algorithm is compared to the *Constant Liar* approach. The paired t-tests show that the proposed EGO–PEI algorithm performs similarly to the *Constant Liar* approach on the Sixhump problem, and performs significantly better than the *Constant Liar* approach on the other five test problems.

In addition, we study the effect on the convergence of the EGO–PEI algorithm when the global optimum of the PEI criterion is not found by the optimizer through numerical experiments. The results show that the performance of the EGO–PEI algorithm is sensitive to the quality of the optimizer of the PEI function. It is recommended to use as much evaluations as one can afford to find the global maximum of the PEI function when using the EGO–PEI algorithm. As further research topics, we would like to test our proposed EGO–PEI algorithm on real world simulation-based optimization problems.

Acknowledgements We would like to thank the anonymous reviewers for their helpful comments.

References

1. Rios, L.M., Sahinidis, N.V.: Derivative-free optimization: a review of algorithms and comparison of software implementations. J. Glob. Optim. **56**(3), 1247–1293 (2013)
2. Boukouvala, F., Misener, R., Floudas, C.A.: Global optimization advances in mixed-integer nonlinear programming, MINLP, and constrained derivative-free optimization, CDFO. Eur. J. Oper. Res. **252**(3), 701–727 (2016)

3. Wang, G.G., Shan, S.: Review of metamodeling techniques in support of engineering design optimization. J. Mech. Des. **129**(2), 370–380 (2007)
4. Viana, F.A.C., Simpson, T.W., Balabanov, V., Toropov, V.: Metamodeling in multidisciplinary design optimization: how far have we really come? AIAA J. **52**(4), 670–690 (2014)
5. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. J. Glob. Optim. **13**(4), 455–492 (1998)
6. Jones, D.R.: A taxonomy of global optimization methods based on response surfaces. J. Glob. Optim. **21**(4), 345–383 (2001)
7. Sasena, M.J., Papalambros, P., Goovaerts, P.: Exploration of metamodeling sampling criteria for constrained global optimization. Eng. Optim. **34**(3), 263–278 (2002)
8. Parr, J.M., Keane, A.J., Forrester, A.I.J., Holden, C.M.E.: Infill sampling criteria for surrogate-based optimization with constraint handling. Eng. Optim. **44**(10), 1147–1166 (2012)
9. Huang, D., Allen, T.T., Notz, W.I., Zeng, N.: Global optimization of stochastic black-box systems via sequential Kriging meta-models. J. Glob. Optim. **34**(3), 441–466 (2006)
10. Forrester, A.I., Keane, A.J., Bressloff, N.W.: Design and analysis of noisy computer experiments. AIAA J **44**(10), 2331–2339 (2006)
11. Knowles, J.: ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. IEEE Trans. Evolut. Comput. **10**(1), 50–66 (2006)
12. Couckuyt, I., Deschrijver, D., Dhaene, T.: Fast calculation of multiobjective probability of improvement and expected improvement criteria for Pareto optimization. J. Glob. Optim. **60**(3), 575–594 (2014)
13. Ginsbourger, D., Le Riche, R., Carraro, L.: Kriging is well-suited to parallelize optimization. In: Tenne, Y., Goh, C.-K. (eds.) Computational Intelligence in Expensive Optimization Problems. Adaptation Learning and Optimization, vol. 2, pp. 131–162. Springer, Berlin (2010)
14. Sóbester, A., Leary, S.J., Keane, A.J.: A parallel updating scheme for approximating and optimizing high fidelity computer simulations. Struct. Multidiscipl. Optim. **27**(5), 371–383 (2004)
15. Feng, Z.W., Zhang, Q.B., Zhang, Q.F., Tang, Q.G., Yang, T., Ma, Y.: A multiobjective optimization based framework to balance the global exploration and local exploitation in expensive optimization. J. Glob. Optim. **61**(4), 677–694 (2015)
16. Forrester, A., Sóbester, A., Keane, A.: Engineering Design Via Surrogate Aodelling: A Practical Guide. Wiley, New York (2008)
17. Bischl, B., Wessing, S., Bauer, N., Friedrichs, K., Weihs, C.: MOI-MBO: multiobjective infill for parallel Mmodel-based optimization. In: Pardalos, P.M., Resende, M.G.C., Vogiatzis, C., Walteros, J.L. (eds.) Learning and Intelligent Optimization. Lecture Notes in Computer Science, pp. 173–186. Springer, Berlin (2014)
18. Hamza, K., Shalaby, M.: A framework for parallelized efficient global optimization with application to vehicle crashworthiness optimization. Eng. Optim. **46**(9), 1200–1221 (2014)
19. Hutter, F., Hoos, H., Leyton-Brown, K.: Parallel algorithm configuration. In: Hamadi, Y., Schoenauer, M. (eds.) Learning and Intelligent Optimization. Lecture Notes in Computer Science, pp. 55–70. Springer, Berlin (2012)
20. Viana, F.A., Haftka, R.T., Watson, L.T.: Efficient global optimization algorithm assisted by multiple surrogate techniques. J. Glob. Optim. **56**(02), 669–689 (2013)
21. Sacks, J., Welch, W.J., Mitchell, T.J., Wynn, H.P.: Design and analysis of computer experiments. Stat. Sci. **4**(4), 409–423 (1989)
22. Torn, A., Zilinskas, A.: Global Optimization. Springer, Berlin (1987)
23. Dixon, L.C.W., Szego, G.P.: The optimization problem: an introduction. In: Dixon, L.C.W., Szego, G.P. (eds.) Towards Global Optimization II. North Holland, New York (1978)
24. Sasena, M.J.: Flexibility and Efficiency Enhancements for Constrained Global Design Optimization with Kriging Approximations. University of Michigan, Ann Arbor (2002)
25. Lophaven, S.N., Nielsen, H.B., Søndergaard, J.: Dace—a matlab Kriging toolbox. Technical Report IMM TR C2002 C12, Technical University of Denmark, Denmark (2002). http://www2.imm.dtu.dk/~hbn/dace/
26. Viana F.A.C.: SURROGATES Toolbox Users Guide. Gainesville, FL, USA, version 3.0 edn. (2011). http://sites.google.com/site/felipeacviana/surrogatestoolbox
27. Price, K.V., Storn, R.M., Lampinen, J.A.: Differential Evolution: A Practical Approach to Global Optimization. Springer, Berlin (2005). http://www1.icsi.berkeley.edu/~storn/code.html
28. Barr, R.S., Hickman, B.L.: Reporting computational experiments with parallel algorithms: issues, measures, and experts' opinions. ORSA J. Comput. **5**(1), 2–18 (1993)
29. Regis, R.G., Shoemaker, C.A.: Parallel radial basis function methods for the global optimization of expensive functions. Eur. J. Oper. Res. **182**(2), 514–535 (2007)