Article

TRANSACTIONS OF THE INSTITUTE OF MEASUREMENT & CONTROL

Transactions of the Institute of Measurement and Control
2015, Vol. 37(2) 190–204

DOI: 10.1177/0142331214537015
tim.sagepub.com
SAGE

Biogeography-based optimization in noisy environments

Haiping Ma[1,2], Minrui Fei[2], Dan Simon[3] and Zixiang Chen[1]

Abstract

Biogeography-based optimization (BBO) is a new evolutionary optimization algorithm that is based on the science of biogeography. In this paper, BBO is applied to the optimization of problems in which the fitness function is corrupted by random noise. Noise interferes with the BBO immigration rate and emigration rate, and adversely affects optimization performance. We analyse the effect of noise on BBO using a Markov model. We also incorporate re-sampling in BBO, which samples the fitness of each candidate solution several times and calculates the average to alleviate the effects of noise. BBO performance on noisy benchmark functions is compared with particle swarm optimization (PSO), differential evolution (DE), self-adaptive DE (SaDE) and PSO with constriction (CPSO). The results show that SaDE performs best and BBO performs second best. In addition, BBO with re-sampling is compared with Kalman filter-based BBO (KBBO). The results show that BBO with re-sampling achieves almost the same performance as KBBO but consumes less computational time.

Keywords

Biogeography-based optimization, evolutionary algorithm, Kalman filter, noisy optimization, re-sampling.

Introduction

Many optimization problems in science and engineering include fitness function noise, which poses a challenge for optimization algorithms (Beyer and Sendhoff, 2007; Hansen et al., 2009; Kheng et al., 2012; Schwefel, 1993; Yu et al., 2008). Noise corrupts the calculation of objective functions via imperfect sensors, measurement devices and approximate numerical simulations. Noise results in two types of undesirable effects in optimization algorithms: 1) a superior candidate solution may erroneously be indicated as inferior, and 2) an inferior candidate solution may erroneously be indicated as superior. These effects result in false optima and reduced optimization performance, including reduced convergence rates and non-monotonic fitness improvement. Evolutionary algorithms (EAs; Chen et al., 2010a) have been modified and applied in several ways to noisy problems (Pietro, 2004). Attractive optimization algorithms for noisy problems include genetic algorithms (GAs; Mühlenbein and Schlierkamp-Voosen, 1993; Stroud, 2001; Yao et al., 1999), estimation of distribution algorithms (EDA; Chen et al., 2010b; Dong et al., 2013), differential evolution (DE; Jin and Branke, 2005; Krink et al., 2004; Liu et al., 2008; Mininno and Neri, 2010), and particle swarm optimization (PSO; Mendel et al., 2011; Pan et al., 2006).

Biogeography-based optimization (BBO; Simon, 2008) is a relatively new EA for global optimization. It is modelled after the immigration and emigration of species between habitats. One distinctive feature of BBO is that in each generation, BBO uses the fitness of each solution to determine its immigration and emigration rate. The emigration rate is proportional to fitness and the immigration rate is inversely proportional to fitness. BBO has demonstrated good performance on benchmark functions (Du et al., 2009; Ergezer et al., 2009; Ma, 2010; Ma and Simon, 2011). It has also been applied to many real-world optimization problems, including sensor selection (Simon, 2008), economic load dispatch (Bhattacharya and Chattopadhyay, 2010), satellite image classification (Panchal et al., 2009), power system optimization (Rarick et al., 2009) and others, but until now, BBO has primarily been applied to deterministic and noiseless optimization problems. The only published report of the use of BBO on noisy problems has been a master's thesis (Du, 2009), which used Kalman filtering (Simon, 2006) to compensate for the effects of noise and to provide a fitness estimate of each candidate solution. The Kalman filter includes the calculation of fitness estimation uncertainty, which increases computational time. Therefore, for many practical optimization problems, this Kalman filter-based BBO might not be viable.

[1]Department of Electrical Engineering, Shaoxing University, Shaoxing, Zhejiang, China
[2]Shanghai Key Laboratory of Power Station Automation Technology, School of Mechatronic Engineering and Automation, Shanghai University, Shanghai, China
[3]Department of Electrical and Computer Engineering, Cleveland State University, Cleveland, Ohio, USA

Corresponding author:
Haiping Ma, Shanghai Key Laboratory of Power Station Automation Technology, School of Mechatronic Engineering and Automation, Shanghai University, Shanghai, China.
Email: Mahp@usx.edu.cn

Previous noise compensation methods in EAs can be classified into two categories (Jin and Branke, 2005): methods that require an increase in computational cost (including explicit averaging methods and implicit averaging methods) and methods that perform hypotheses testing on the noise (including the use of approximate fitness models and the modification of selection schemes). Explicit averaging methods include re-sampling (Krink et al., 2004; Pietro et al., 2004), which is the most common approach to dealing with noise. Re-sampling of the fitness values involves several noisy fitness value measurements, followed by averaging to obtain an improved fitness estimate. Averaging an increased number of samples reduces the variance of the estimated fitness. As the number of samples increases to infinity, the uncertainty in the fitness estimate decreases to zero, which transforms the noisy problem into a noiseless one.

Variants of re-sampling include dynamic re-sampling, standard error dynamic re-sampling and m-level re-sampling (Pietro et al., 2004). Re-sampling has limitations because it leads to an increase in the number of fitness evaluations, which means that computational time increases, but compared with the more complex calculations of the Kalman filter, re-sampling is simpler and faster, as we show in this paper.

Implicit averaging methods increase the population size so that candidate solutions can be re-evaluated during the normal course of evolution, and so that neighbouring solutions can be evaluated, which gives fitness estimates in neighbouring regions of the search space. It has been shown in Fitzpatrick and Grefenstette (1988) that a large population size reduces the influence of noise on the optimization process. The main idea of approximated model methods is that measured fitness values of neighbouring individuals can give good fitness estimates without extra evaluations (Neri et al., 2008).

The aim of this paper is to study the performance of BBO on the optimization of noisy problems, and to study the effect of noise on BBO immigration and emigration rates. We use a Markov model to analyse the effect of noise on BBO, and then we incorporate re-sampling in BBO to alleviate the effects of noise. The methods in this paper could also be extended to other EAs in future work.

The original contributions of this paper include the following: 1) We use a Markov model to mathematically analyse the effect of fitness function evaluation noise on BBO performance. We find that higher mutation rates tend to reduce the effect of noise on BBO performance, although higher mutation rates might themselves reduce BBO performance. 2) EA performance on noisy fitness function benchmarks, in order from best to worst, is self-adaptive DE (SaDE), BBO and PSO with constriction (CPSO), DE and PSO. 3) BBO with re-sampling performs as well as Kalman filter-based BBO on noisy optimization problems, but with a lower computational cost.

The remainder of this paper is organized as follows. The next section reviews BBO and its Markov model, then we use the Markov model to analyse the influence of noise on BBO. We present performance comparisons between BBO, PSO, DE, CPSO and SaDE on noisy benchmark functions, then we provide comparisons between BBO with re-sampling and Kalman filter-based BBO. Lastly, we present conclusions and suggest directions for future work.

Natural biogeography and biogeography-based optimization

This section presents an overview of natural biogeography, an overview of standard BBO and an overview of a previously derived Markov model for BBO.

Natural biogeography

Biogeography is nature's way of distributing species, and it has often been studied as a process that maintains equilibrium in natural habitats. Species equilibrium in a biological habitat occurs when the combined speciation and immigration rates equals the extinction rate. One reason that biogeography has been viewed from the equilibrium perspective is that this viewpoint was the first to place biogeography on a firm mathematical footing (MacArthur and Wilson, 1963, 1967). However, since then, the equilibrium perspective has been increasingly questioned, or rather expanded, by biogeographers.

In engineering, we often view stability and optimality as competing objectives; for example, a simple system is typically easier to stabilize than a complex system, while an optimal system is typically more complex and less stable than a simpler system (Keel and Bhattacharyya, 1997). However, in biogeography, stability and optimality are two perspectives of the same phenomenon. Optimality in biogeography involves biologically diverse, complex communities that are highly adaptable to their environment. Stability in biogeography involves the persistence of existing populations. Field observations show that complex communities are more adaptable and stable than simple communities (Harding, 2006: 82), and this observation has been supported by simulation (Elton, 1958; MacArthur, 1955). The equilibrium versus optimality debate in biogeography thus becomes a matter of semantics; equilibrium and optimality are simply two different views of the same behaviour.

Some examples of biogeography as an optimization process are the migration of species to Krakatoa, a volcanic island in the Indian Ocean, which erupted in 1883 (Whittaker and Bush, 1993); the Amazon rainforest, which is a typical case of a mutually optimizing life/environment system (Harding, 2006); Earth's temperature (Harding, 2006); Earth's atmospheric composition (Lenton, 1998); and the ocean's mineral content (Lovelock, 1990). This is not to say that biogeography is optimal for any particular species. Life flourishes and evolves on Earth, but not necessarily in a human-centric way.

Biogeography is a positive feedback phenomenon, similar to natural selection. In natural selection, as species become fitter, they are more likely to survive. As they thrive, they disperse and become better able to adapt to their environment. Natural selection, like biogeography, entails positive feedback. However, the time scale of biogeography is much shorter than that of natural selection, which hints at the possibility of improved optimization performance by using biogeography rather than natural selection as a motivating paradigm for optimization (i.e. BBO rather than GAs). The viewpoint of biogeography as an optimization process motivated the development of BBO as an evolutionary optimization algorithm (Simon, 2008), which we discuss next.

Biogeography-based optimization

BBO is a new optimization approach inspired by biogeography. A biogeography habitat corresponds to a *candidate solution* of an optimization problem. Therefore, the number of habitats in BBO corresponds to the BBO population size. Each candidate solution is comprised of a set of features, which are similar to *genes* in GAs, and which are also called independent variables or decision variables. The number of species in each habitat corresponds to the problem dimension. We see that contrary to natural biogeography, all of the habitats in BBO (i.e. the candidate solutions) have the same number of species (i.e. independent variables).

Like other EAs (Schwefel, 1993), BBO probabilistically shares information between candidate solutions to improve candidate solution fitness. In BBO, each candidate solution immigrates features from other candidate solutions based on its immigration rate, and emigrates features to other candidate solutions based on its emigration rate. In the original BBO paper (Simon, 2008), immigration rates are first used to decide probabilistically whether to immigrate solution features to a given solution. Then, if immigration is selected, emigration rates are used to choose the emigrating solution. Migration can be expressed as

$$x_i(s) \leftarrow x_j(s) \tag{1}$$

where x_i denotes the immigrating solution, x_j denotes the emigrating solution and s denotes a solution feature index. In BBO, each candidate solution x has an immigration rate λ and emigration rate μ. A good solution has relatively high μ and low λ, while the converse is true for a poor solution. According to Simon (2008), these functions can be calculated as

$$\begin{aligned} \lambda &= 1 - f(x) \\ \mu &= f(x) \end{aligned} \tag{2}$$

where f denotes solution fitness and is normalized to the range [0, 1]. After migration, we probabilistically decide whether to mutate each feature of each candidate solution.

A description of one generation of BBO is given in Algorithm 1. Migration and mutation of the entire population takes place before any of the solutions are replaced in the population, which requires the use of the temporary population z in the algorithm. In Algorithm 1, the statement 'use λ_k to probabilistically decide whether to immigrate to z_k' can be implemented with the following logic, where rand(0, 1) is a random number uniformly distributed between 0 and 1:

```
If λ_k < rand(0,1) then
  Immigration to z_k does occur
else
  Immigration does not occur
end if
```

In Algorithm 1, the statement 'Use $\{\mu_i\}$ to probabilistically select the emigrating solution y_j' can be implemented with any fitness-based selection method since μ_i is proportional to the fitness of y_i. For instance, we could use tournament selection by randomly choosing two or more solutions for a tournament, and then selecting y_j as the fittest solution in the tournament. In this paper, as in most other BBO implementations, we use $\{\mu_i\}$ in a roulette-wheel algorithm so that the probability that each individual y_i is selected for emigration is proportional to its emigration rate μ_i. Standard BBO uses rank-based selection, i.e. we rank the individuals according to fitness values, giving the best individual a rank of N (where N is the population size), and giving the worst individual a rank of 1. Rank-based selection then assigns λ and μ on the basis of rankings rather than on the basis of absolute fitness values (Simon, 2013).

A Markov model of BBO

This section reviews a Markov model of BBO. This model will be used later to mathematically analyse the effect of fitness function noise on BBO.

Algorithm 1: One generation of the BBO algorithm, where N is the population size. y is the entire population of candidate solutions, y_k is the kth candidate solution, and $y_k(s)$ is the sth feature of y_k.

```
For each solution y_k, define emigration rate μ_k proportional to fitness of y_k, where μ_k∈[0,1]
For each solution y_k, define immigration rate λ_k = 1 − μ_k
z←y
For each solution z_k (k = 1 to N)
  For each solution feature s
    Use λ_k to probabilistically decide whether to immigrate to z_k
    If immigrating then
      Use {μ_i} to probabilistically select the emigrating solution y_j
      z_k(s) ←y_j(s)
    End if
  Next solution feature
  Probabilistically decide whether to mutate z_k
Next solution
y←z
```

Consider a q-dimensional binary optimization problem with search space $\{x_1, \ldots, x_n\}$. The search space is the set of all bit strings x_i, each consisting of q bits. Therefore, the cardinality of the search space is $n = 2^q$. Suppose that BBO is currently in generation t. Denote the kth candidate solution in the BBO population as y_k, where $k \in [1, N]$, and where N is the population size. Based on the previously derived transition probability of BBO (Simon et al., 2011), the probability that migration results in y_k being equal to x_i at generation $t + 1$, is given by

$$m_{ki} = \Pr(y_{k,t+1} = x_i) = \prod_{s=1}^{q} \left[\underbrace{((1-\lambda_k)\mathbf{1}_0(y_k(s) - x_i(s)))}_{\substack{\text{Probability if immigration} \\ \text{does not occur}}} + \underbrace{\left(\lambda_k \frac{\sum_{j \in \varsigma_i(s)} v_j \mu_j}{\sum_{j=1}^{n} v_j \mu_j} \right)}_{\substack{\text{Probability if immigration} \\ \text{does occur}}} \right] \tag{3}$$

where $\mathbf{1_0}$ is the indicator function on the set 0 (i.e. $\mathbf{1_0}(a) = 1$ if $a = 0$, and $\mathbf{1_0}(a) = 0$ if $a \neq 0$), s denotes the index of the candidate solution feature (i.e. the bit number), λ_k denotes the immigration rate of candidate solution y_k, μ_j denotes the emigration rate of the candidate solution x_j and v_j denotes the number of x_j individuals in the population. The notation $\varsigma_i(s)$ denotes the set of search space indices j such that the sth bit of x_j is equal to the sth bit of x_i, i.e. $\varsigma_i(s) = \{j : x_j(s) = x_i(s)\}$. m_{ki} is the probability that the kth individual in the population is equal to the ith individual in the search space when only migration is considered (no mutation). Note that the first term in the product on the right side of (3) denotes the probability that $y_{k,t+1}(s) = x_i(s)$ if immigration of the sth candidate solution feature did not occur, and the second term denotes the probability if immigration of the sth candidate solution feature did occur. For a detailed derivation, see Simon et al. (2011).

Mutation operates independently on each candidate solution by probabilistically reversing each bit in each candidate solution. Suppose that the event that each bit of a candidate solution is flipped is stochastically independent and occurs with probability $p_m \in (0, 1)$. Then the probability that a candidate solution that is equal to x_i mutates to x_j can be written as

$$u_{ij} = \Pr(x_i \to x_j) = p_m^{H_{ij}} (1 - p_m)^{q - H_{ij}} \tag{4}$$

where q is the number of bits in each candidate solution, and H_{ij} represents the Hamming distance between bit strings x_i and x_j.

The Markov model presented above will be used in the following section to analyse mathematically the effect of noise on BBO.

The influence of noise on BBO

Up to this point, BBO applications in the literature have typically been implemented on deterministic problems. That means that the fitness calculation of each solution is noise-free, but in the real world, noiseless environments do not exist. In a noisy environment, the calculated fitness is not equal to the true fitness, the immigration and emigration rates in BBO will be calculated incorrectly, and BBO migration may not accomplish its intended purpose. Fitness noise can be represented in a very general form (Jin and Branke, 2005), but in this paper we assume the most simple and most common type of noise, which is additive and Gaussian.

Consider two solutions, x_1 and x_2. Their true fitness values are denoted by f_1 and f_2, respectively, and the fitness function evaluation noise is denoted by w_1 and w_2, respectively. Assume that the true fitness of x_1 is better than that of x_2, i.e.

$$f_1 > f_2 \tag{5}$$

However, the measured fitness of x_1 may be less than that of x_2 because of noise, i.e.

$$f_1 + w_1 < f_2 + w_2 \tag{6}$$

If noise has a strong effect on measured fitness, the rank of the measured fitness values could be much different from the rank of the true fitness values. Du (2009) computes the probability of fitness ranks changing due to noise.

In this paper, we assume w_1 and w_2 are additive noises, because additive noise is the predominant noise model due to its frequent occurrence in various measurement systems. Additive noise is often assumed to be Gaussian due to its wide prevalence in both natural and engineering systems. Non-Gaussian noise, such as Cauchy noise, has also been considered (Arnold and Beyer, 2003). It is plausible to assume that the noise cannot exceed certain limits due to the characteristics of the fitness measurement instrument. These assumptions have theoretical and practical impacts on noisy EAs, but are not considered further in this paper.

Example

Now we give an example of the effect of noise on BBO performance using the Markov transition probabilities of the previous section. Suppose we have a two-bit problem ($q = 2$, $n = 4$) with a population size $N = 3$. The search space consists of bit strings $x = \{x_1, x_2, x_3, x_4\} = \{00, 01, 10, 11\}$ with corresponding fitness values $f = \{f_1, f_2, f_3, f_4\} = \{0.2, 0.4, 0.6, 0.8\}$. Suppose that the three individuals in the current population are $y = \{x_1, x_2, x_3\} = \{00, 01, 10\}$. In the noise-free case, the fitness value of x_1 is $f_1 = 0.2$, and its corresponding immigration rate and emigration rate are $\lambda_1 = 0.8$ and $\mu_1 = 0.2$, as indicated by (2). The fitness value of x_2 is $f_2 = 0.4$, with corresponding immigration rate and emigration rate $\lambda_2 = 0.6$ and $\mu_2 = 0.4$. We perform probabilistic migration to see if x_1 and x_2 can transition to the optimal solution $x_4 = 11$. Based on (3), the probability of x_1 transitioning to the optimal solution due to migration only (no mutation) is

$$\begin{aligned}
&\Pr(x_1 \to x_4) \\
&= \prod_{s=1}^{2}\left[((1-\lambda_1)\mathbf{1}_0(x_1(s)-x_4(s))) + \left(\lambda_1 \frac{\sum_{j\in \varsigma_4(s)} v_j\mu_j}{\sum_{j=1}^{4} v_j\mu_j}\right)\right] \\
&= \left[(1-\lambda_1)\mathbf{1}_0(x_1(1)-x_4(1)) + \lambda_1 \frac{\sum_{j\in\{3,\,4\}} v_j\mu_j}{\sum_{j=1}^{4} v_j\mu_j}\right] \\
&\quad \left[(1-\lambda_1)\mathbf{1}_0(x_1(2)-x_4(2)) + \lambda_1 \frac{\sum_{j\in\{2,\,4\}} v_j\mu_j}{\sum_{j=1}^{4} v_j\mu_j}\right] \\
&= 0.107
\end{aligned}$$

The probability of x_2 transitioning to the optimal solution due to migration only is

$$\begin{aligned}
&\Pr(x_2 \to x_4) \\
&= \prod_{s=1}^{2}\left[((1-\lambda_2)\mathbf{1}_0(x_2(s)-x_4(s))) + \left(\lambda_2 \frac{\sum_{j\in \varsigma_4(s)} v_j\mu_j}{\sum_{j=1}^{4} v_j\mu_j}\right)\right] \\
&= \left[(1-\lambda_2)\mathbf{1}_0(x_2(1)-x_4(1)) + \lambda_2 \frac{\sum_{j\in\{3,\,4\}} v_j\mu_j}{\sum_{j=1}^{4} v_j\mu_j}\right] \\
&\quad \left[(1-\lambda_2)\mathbf{1}_0(x_2(2)-x_4(2)) + \lambda_2 \frac{\sum_{j\in\{2,\,4\}} v_j\mu_j}{\sum_{j=1}^{4} v_j\mu_j}\right] \\
&= 0.180
\end{aligned}$$

Next suppose that noise corrupts the measured fitness of x_1 and x_2. Suppose that the measured fitness of x_1 is $f'_1 = 0.3$ and the measured fitness of x_2 is $f'_2 = 0.2$, so that $f'_1 > f'_2$. In this case, the immigration rate and the emigration rate of x_1 are $\lambda'_1 = 0.7$ and $\mu'_1 = 0.3$ respectively, and the immigration rate and the emigration rate of x_2 are $\lambda'_2 = 0.8$ and $\mu'_2 = 0.2$ respectively. We perform a migration trial to see if x_1 and x_2 can transition to the optimal solution $x_4 = 11$. Based on (3), the probability of x_1 transitioning to the optimal solution due to migration only is

$$\begin{aligned}
&\Pr_{\text{noise}}(x_1 \to x_4) \\
&= \prod_{s=1}^{2}\left[((1-\lambda_1)\mathbf{1}_0(x_1(s)-x_4(s))) + \left(\lambda_1 \frac{\sum_{j\in \varsigma_4(s)} v_j\mu_j}{\sum_{j=1}^{4} v_j\mu_j}\right)\right] = 0.049
\end{aligned}$$

The probability of x_2 transitioning to the optimal solution due to migration only is

$$\begin{aligned}
&\Pr_{\text{noise}}(x_2 \to x_4) \\
&= \prod_{s=1}^{2}\left[((1-\lambda_2)\mathbf{1}_0(x_2(s)-x_4(s))) + \left(\lambda_2 \frac{\sum_{j\in \varsigma_4(s)} v_j\mu_j}{\sum_{j=1}^{4} v_j\mu_j}\right)\right] = 0.151
\end{aligned}$$

We see that the probabilities that the two individuals x_1 and x_2 transition to the optimal solution change significantly. We further find that these two probabilities both decrease, with the probability of x_1 decreasing from 0.107 to 0.049, and the probability of x_2 decreasing from 0.180 to 0.151.

Now suppose that the mutation rate probability p_m is 0.1 per bit. We can combine (3) and (4) to find the following transition probabilities in the noise-free case:

$$\begin{aligned}
\Pr_m(x_1 \to x_4) &= 0.132 \\
\Pr_m(x_2 \to x_4) &= 0.197 \\
\Pr_{m,\,\text{noise}}(x_1 \to x_4) &= 0.082 \\
\Pr_{m,\,\text{noise}}(x_2 \to x_4) &= 0.169
\end{aligned}$$

We see that even with mutation, the probability of transitioning to the optimal solution x_4 decreases when noise corrupts the fitness evaluations. However, mutation tends to even out the probabilities. Without mutation, we saw that the probability of x_1 transitioning to the optimal solution decreases from 0.107 to 0.049, a decrease of 54%, and the probability of x_2 transitioning to the optimal solution decreases from 0.180 to 0.151, a decrease of 16%. However, with a mutation rate of 0.1, we saw that the probability of x_1 transitioning to the optimal solution decreases from 0.132 to 0.082, a decrease of 38%; and the probability of x_2 transitioning to the optimal solution decreases from 0.197 to 0.169, a decrease of 14%. Noise damages the migration mechanism of BBO, but some of that damage can be mitigated with a high mutation rate.

Simulation results

In this section, we apply re-sampling to BBO to improve BBO performance in noisy environments. We also compare BBO with other EAs that use re-sampling, including PSO, DE, CPSO and SaDE. Then we compare the performance of BBO with re-sampling and BBO augmented with Kalman filtering (KBBO).

BBO with re-sampling

In noisy problems, measured fitness values include noise. Therefore, as we showed in the previous section, the measured values are not perfectly accurate, and they do not perfectly reflect the true value of the fitness. BBO uses the fitness of each solution to determine its immigration and emigration rate. Because of noise, measured fitness is not true fitness, the immigration and emigration rates in BBO are incorrect, and this negatively affects BBO migration. Re-sampling is used to sample the fitness of each candidate solution several times and calculate the average as the estimated fitness.

Suppose that the ith sample $g_i(x)$ of the fitness function of a candidate solution x is given by

$$g_i(x) = f(x) + w_i \tag{7}$$

where $f(x)$ is the true fitness, and w_i is the additive noise at the ith measurement. If we re-sample the measured fitness function l times, the best estimate of the true fitness is

$$\hat{f}(x) = \frac{1}{l}\sum_{i=1}^{l} g_i(x) \tag{8}$$

Re-sampling is a straightforward and effective way to handle noise in fitness functions, and one of the most important contributions of re-sampling is that it does not need any control parameters except for the number of re-samples. The flowchart of BBO with re-sampling is shown in Figure 1. It is

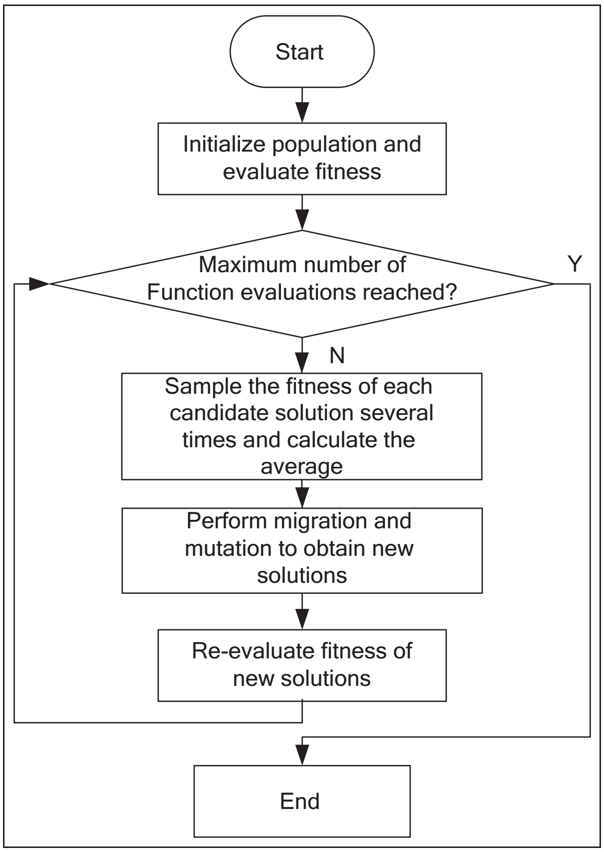

Figure 1. Flowchart of biogeography-based optimization with re-sampling.

worth pointing out that in Figure 1, we can use PSO, DE or any other EA instead of BBO to alleviate the effects of noise.

Test set-up

In this paper, we use a fixed number of total fitness evaluations for each benchmark and each algorithm to provide fair performance comparisons. We run experiments with $l=1$, 5, 20 and 50, where l is the total number of fitness evaluations per candidate solution per generation. We also compare performance results on noise-free problems.

A representative set of noiseless and noisy benchmark functions have been used for performance testing. For the noiseless functions, we have selected the 25 test problems, each with 30 dimensions, which appeared in the CEC 2005 special session on real parameter optimization (Suganthan et al., 2005). These noiseless functions are summarized in Table 1, and include five unimodal functions and 20 multimodal functions. The functions also include 12 basic functions, two expanded functions and 11 hybrid functions. Many other benchmarks have been published in the literature, but we use these benchmarks because many studies of EA performance on these benchmarks are available in the literature.

All functions are shifted in order to ensure that their optima are not at the centre of the search space. The noisy benchmark functions are defined as

$$f_{Noisy}(\vec{x}) = f(\vec{x}) + |N(0,\ 1)| \tag{9}$$

where $|N(0,\ 1)|$ is the absolute value of a Gaussian random variable with mean 0 and variance 1. Note that all benchmark functions are minimization problems.

Table 1. Benchmark functions.

Function	Name	Domain	Minimum
F1	Shifted Sphere Function	$-100 \leqslant x_i \leqslant 100$	−450
F2	Shifted Schwefel Problem 1.2	$-100 \leqslant x_i \leqslant 100$	−450
F3	Shifted Rotated High Conditioned Elliptic Function	$-100 \leqslant x_i \leqslant 100$	−450
F4	Shifted Schwefel Problem 1.2 with Noise in Fitness	$-100 \leqslant x_i \leqslant 100$	−450
F5	Schwefel Problem 2.6 with Global Optimum on Bounds	$-100 \leqslant x_i \leqslant 100$	−310
F6	Shifted Rosenbrock Function	$-100 \leqslant x_i \leqslant 100$	390
F7	Shifted Rotated Griewank Function without Bounds	$0 \leqslant x_i \leqslant 600$	−180
F8	Shifted Rotated Ackley's Function with Global Optimum on Bounds	$-32 \leqslant x_i \leqslant 32$	−140
F9	Shifted Rastrigin Function	$-5 \leqslant x_i \leqslant 5$	−330
F10	Shifted Rotated Rastrigin Function	$-5 \leqslant x_i \leqslant 5$	−330
F11	Shifted Rotated Weierstrass Function	$-0.5 \leqslant x_i \leqslant 0.5$	90
F12	Schwefel Problem 2.13	$-100 \leqslant x_i \leqslant 100$	−460
F13	Expanded Extended Griewank plus Rosenbrock Function (F8F2)	$-3 \leqslant x_i \leqslant 1$	−130
F14	Shifted Rotated Expanded Schaffer F6	$-100 \leqslant x_i \leqslant 100$	−300
F15	Hybrid Composition Function	$-5 \leqslant x_i \leqslant 5$	120
F16	Rotated Hybrid Composition Function	$-5 \leqslant x_i \leqslant 5$	120
F17	Rotated Hybrid Composition Function with Noise in Fitness	$-5 \leqslant x_i \leqslant 5$	120
F18	Rotated Hybrid Composition Function	$-5 \leqslant x_i \leqslant 5$	10
F19	Rotated Hybrid Composition Function with a Narrow Basin for the Global Optimum	$-5 \leqslant x_i \leqslant 5$	10
F20	Rotated Hybrid Composition Function with the Global Optimum on the Bounds	$-5 \leqslant x_i \leqslant 5$	10
F21	Rotated Hybrid Composition Function	$-5 \leqslant x_i \leqslant 5$	360
F22	Rotated Hybrid Composition Function with high Condition Number matrix	$-5 \leqslant x_i \leqslant 5$	360
F23	Non-Continuous Rotated Hybrid Composition Function	$-5 \leqslant x_i \leqslant 5$	360
F24	Rotated Hybrid Composition Function	$-5 \leqslant x_i \leqslant 5$	260
F25	Rotated Hybrid Composition Function without Bounds	$-5 \leqslant x_i \leqslant 5$	260

More details about these functions can be found in Suganthan et al. (2005).

Comparisons with other EAs

To illustrate the performance BBO on noisy optimization problems, we compare with four other EAs: a basic DE algorithm, a basic PSO algorithm, SaDE and CPSO. All algorithms are combined with re-sampling. Note that the four algorithms that we choose form a representative set rather than a complete set. We compare with DE because it is an effective EA and has demonstrated excellent performance (Das and Suganthan, 2011). We compare with SaDE because it is one of the best DE variants (Zhao et al., 2011), and it uses a self-adaptive mechanism on control parameters F (scaling factor) and CR (crossover rate); each candidate solution in the population is extended with control parameters F and CR that are adjusted during evolution. We compare with the current standard PSO algorithm obtained from Particle Swarm Central (http://www.particleswarm.info/) because it usually offers good performance and is a relatively new EA (Bratton and Kennedy, 2007). We compare with CPSO because it has a structure that is more complex than standard PSO, and demonstrates good performance (Clerc and Kennedy, 2002; Eberhart and Shi, 2000).

For BBO, the following parameters have to be tuned: population size, maximum migration rate and mutation rate. In Ma (2010), these parameters have been discussed in detailed. Here we use a reasonable set of tuning parameters, but do not make any effort at finding the best settings. The parameters that we use are: maximum immigration and emigration rate of 1, and mutation probability of 0.01 per generation per solution feature, with uniform mutation centred at the middle of the search domain. In addition, we use linear migration curves as described in (2).

For DE, we use the parameters recommended by Clerc (2006), Eberhart and Shi (2004), Eberhart et al. (2001), and Onwubolu and Babu (2004): F = 0.5 and CR = 0.5.

For PSO, we use the parameters recommended by Onwubolu and Babu (2004), Price and Storn (1997), and Storn (1999): inertia weight of 0.3, cognitive constant of 1, social constant for swarm interaction of 1.0, and social constant for neighbourhood interaction of 1.0.

For SaDE, the parameter settings are adapted according to the learning progress (Zhao et al., 2011): F is randomly sampled from the normal distribution N(0.5, 0.3), and CR follows the normal distribution N(0.5, 0.1).

For CPSO, we use a constriction coefficient $K = 2\Big/\left|2-\varphi-\sqrt{\varphi^2-4\varphi}\right|$, where $\varphi = 4.1$ (Clerc and Kennedy, 2002). The other parameters of CPSO are the same as those of PSO.

Each algorithm has a population size of 50, and a fixed number of total fitness evaluations (NumEval) of 100,000. The noise-handling method in each algorithm uses re-sampling. Each function is optimized over 25 independent runs. We use the same set of initial random populations to evaluate each algorithm.

The benchmark function results are shown in Tables 2–4. We observe that for noiseless benchmark functions, SaDE performs the best on 14 functions, BBO performs the best on six functions, CPSO performs the best on five functions, DE performs the best on one function and PSO does not perform best on any of the functions.

The number of fitness re-samples strongly affects optimization performance. When the number of fitness re-samples is one, SaDE performs the best on 14 functions, CPSO performs the best on four functions, DE performs the best on four functions, PSO performs the best on three functions and BBO performs the best on two functions. When the number of fitness re-samples is five, 20 or 50, SaDE performs the best on 14 functions, BBO performs the best on six functions, CPSO performs the best on five functions, DE performs the best on one function and PSO does not perform best on any of the functions. Note that these results are the almost same as those obtained for the noiseless benchmark functions. If we tested other state-of-the-art DE and PSO algorithms, we could probably obtain better optimization results (Das et al., 2005; Mendel et al., 2011; Pietro et al., 2004). However, the same could be said for recently proposed improvements of BBO (Du et al., 2009; Ergezer et al., 2009).

We find that for many of the noisy benchmark functions, the performance of BBO does not dramatically improve as the number of re-samples increases. For example, for F3, BBO performs almost the same when the number of fitness re-samples is 20 or 50, but worse than when the number of fitness re-samples is five. There is a point of diminishing returns with the number of re-samples. If the number of re-samples is too large, then we end up wasting fitness function evaluations on increased estimation accuracy, because we already have sufficient estimation accuracy with fewer re-samples.

We see that re-sampling alleviates the effect of noise for the benchmark functions that we studied, but it is hard to quantify how many times we have to sample a noisy function to achieve a desired fitness value. Many re-samples might not be feasible for expensive fitness functions, and might not be necessary for all problems. However, our results show that re-sampling is a simple and effective approach to deal with noise.

Table 5 shows the results of Wilcoxon test comparisons between BBO and each of the other four EAs. The Wilcoxon test is a non-parametric statistical method to determine whether differences between groups of data are statistically significant when the assumptions that the differences are independent and identically normally distributed are not satisfied (Al-Rifaie and Blackwell, 2012; Demsar, 2006; Derrac et al., 2011). Pairs are marked in Table 5 if the difference between the pair of algorithms has a level of significance $\alpha = 0.05$ or less. We have a total 125 groups of data for the noiseless and noisy benchmark functions. We see the following from Table 5:

- There are 63 statistically significant differences between DE and BBO, including:
  - 37 groups of data for which BBO is better;
  - 26 groups of data for which DE is better.
- There are 90 statistically significant differences between SaDE and BBO, including:
  - 31 groups of data for which BBO is better;
  - 59 groups of data for which SaDE is better.
- There are 75 statistically significant differences between PSO and BBO, including:
  - 61 groups of data for which BBO is better;
  - 14 groups of data for which PSO is better.

Table 2. Simulation results for F1–F8.

Function	DE	SaDE	PSO	CPSO	BBO
F1	2.23E−08±4.16E−09	3.76E−08±2.87E−09	3.28E+01±6.77E+00	7.20E−08±5.26E−09	**0.00E+00±0.00E+00**
F1*, *l*=1	**0.00E+00±0.00E+00**	8.63E−04±2.19E−05	7.12E+01±4.19E+00	3.19E−05±4.48E−06	2.45E−01±3.46E−02
F1*, *l*=5	4.91E−02±5.20E−03	5.76E−04±1.34E−05	2.47E+01±1.85E+00	5.44E−01±2.25E−02	**3.16E−06±7.04E−07**
F1*, *l*=20	5.70E+00±1.66E−01	2.21E−01±1.99E−02	2.71E+01±1.99E+00	6.78E+00±1.35E−01	**3.47E−05±2.35E−06**
F1*, *l*=50	9.54E+00±6.34E−01	6.98E−01±3.22E−02	3.22E+01±1.57E+00	2.90E+00±1.27E−01	**7.19E−04±4.65E−05**
F2	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**	7.89E+00±1.54E−01	1.72E−02±3.59E−03	4.11E−03±2.35E−04
F2*, *l*=1	**0.00E+00±0.00E+00**	6.78E−04±0.00E−05	1.38E+01±5.26E+00	7.66E−02±8.21E−03	7.84E−02±3.22E−03
F2*, *l*=5	**7.03E−04±1.29E−05**	8.45E−04±8.26E−03	7.11E+01±3.48E+00	6.98E−02±1.32E−03	5.24E−02±8.15E−03
F2*, *l*=20	**7.58E−02±3.47E−03**	9.64E−02±2.95E−03	6.01E+01±3.05E+00	4.27E−01±5.11E+00	6.34E−01±2.18E−02
F2*, *l*=50	**8.17E−02±7.65E−03**	8.90E−02±3.45E−03	9.12E+01±7.41E+00	2.45E−01±3.31E+00	7.45E−01±5.33E−02
F3	5.34E−07±6.15E−08	**2.51E−09±8.90E−10**	7.02E+02±4.96E+01	4.43E−01±1.78E−02	9.04E+00±1.28E−01
F3*, *l*=1	4.00E−06±3.24E−07	**1.44E−07±1.36E−08**	5.93E+03±1.22E+01	5.29E+00±3.74E−01	7.64E+02±8.23E+01
F3*, *l*=5	8.14E+01±9.25E+00	**2.55E+00±1.17E+00**	7.65E+03±2.39E+01	2.58E+01±4.36E+00	9.05E+01±2.34E+00
F3*, *l*=20	7.34E+01±5.27E+00	**6.60E+00±7.19E−01**	1.46E+03±2.11E+01	1.18E+01±3.60E+00	2.62E+02±7.51E+01
F3*, *l*=50	3.20E+01±5.63E−01	**2.43E+00±8.09E−01**	7.81E+03±1.66E+02	1.09E+01±2.98E−01	2.24E+02±8.47E+01
F4	9.30E−08±1.24E−09	5.67E−09±2.85E−10	2.72E+00±5.34E+00	**1.44E−09±7.86E−10**	6.04E−09±4.45E−10
F4*, *l*=1	**9.71E−14±7.56E−15**	4.36E−07±1.18E−08	3.02E+01±2.33E+00	2.19E−08±5.48E−09	1.78E+00±5.47E−01
F4*, *l*=5	8.04E−01±3.14E−00	6.97E−05±5.19E−06	7.64E+01±8.23E+00	**2.36E−06±6.62E−07**	9.38E−03±2.55E−04
F4*, *l*=20	4.36E−01±2.18E−00	2.44E−03±6.31E−04	7.65E+01±9.04E+00	**1.99E−04±5.47E−05**	1.24E−03±3.65E−04
F4*, *l*=50	8.16E−01±2.58E−00	9.08E−03±1.16E−04	1.37E+01±8.42E+00	**2.19E−04±3.52E−05**	5.69E−03±3.22E−04
F5	4.78E−02±2.54E−03	**4.76E−03±1.23E−04**	7.68E+02±1.23E+01	6.34E+01±7.08E+00	4.57E+00±2.74E−01
F5*, *l*=1	8.24E−01±5.17E−02	**9.90E−02±1.35E−03**	8.65E+02±4.18E+01	5.10E−01±4.52E−02	9.07E+02±5.22E+01
F5*, *l*=5	9.26E+01±1.72E+00	**8.03E+00±2.29E+00**	7.44E+02±2.35E+01	6.90E+01±2.63E+00	9.02E+01±7.31E+00
F5*, *l*=20	2.58E+01±4.16E+00	**5.71E+00±2.58E−01**	7.17E+02±3.90E+01	1.97E+01±2.54E+00	4.66E+01±6.01E+00
F5*, *l*=50	6.71E+01±3.58E+00	**4.67E+00±7.70E−01**	9.01E+02±9.87E+01	2.16E+01±8.09E+00	5.74E+02±5.36E+01
F6	1.28E−08±4.56E−09	3.26E−10±1.19E−11	9.32E+02±5.44E+01	3.88E−02±4.32E−03	**0.00E+00±0.00E+00**
F6*, *l*=1	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**	9.00E+03±1.12E+01	8.36E+00±4.29E−01	4.92E+01±1.37E+00
F6*, *l*=5	9.12E−06±7.65E−07	4.54E−04±2.21E−05	8.32E+03±7.45E+02	7.86E−01±1.10E−02	**9.04E−06±4.32E−07**
F6*, *l*=20	5.86E−06±4.25E−07	1.97E−03±2.53E−04	4.01E+03±2.36E+02	5.27E−01±8.09E−02	**7.69E−07±5.31E−08**
F6*, *l*=50	7.13E−06±2.48E−07	9.02E−03±3.11E−04	5.27E+03±4.17E+02	1.66E−01±2.14E−02	**0.00E+00±0.00E+00**
F7	9.04E+03±3.12E+02	5.65E+02±9.65E+01	8.54E+02±8.77E+01	6.87E+02±9.12E+01	**1.46E+01±3.54E+00**
F7*, *l*=1	5.21E+02±2.47E+01	3.16E+02±5.44E+01	**1.00E+01±2.74E+00**	8.43E+02±6.77E+01	9.74E+03±2.53E+04
F7*, *l*=5	6.35E+03±7.98E+02	1.92E+02±3.31E+01	3.72E+01±4.26E+00	**5.47E+00±3.16E+00**	8.73E+03±2.17E+04
F7*, *l*=20	1.23E+02±5.74E+01	4.38E+01±4.67E+00	8.34E+01±2.47E+00	**4.40E+00±6.49E−01**	5.68E+04±1.24E+05
F7*, *l*=50	2.31E+03±4.06E+02	2.16E+02±3.88E+01	3.54E+01±1.28E+00	**8.81E+00±2.27E−01**	3.72E+04±4.26E+05
F8	5.20E+01±3.98E+00	1.98E−02±2.54E−03	9.60E+02±8.46E+01	**6.78E−03±1.15E−04**	2.74E−02±4.96E−03
F8*, *l*=1	1.38E+00±7.14E−01	7.74E+00±3.29E−01	7.72E+01±1.09E+00	3.76E+00±1.23E−01	**1.05E−01±2.74E−02**
F8*, *l*=5	7.54E+01±2.16E−01	6.89E+01±1.36E+01	7.38E+02±4.47E+01	9.07E+01±7.44E+00	**5.69E+00±4.48E−01**
F8*, *l*=20	5.36E+01±4.18E+00	5.81E+01±2.55E+00	8.92E+02±1.05E+01	1.12E+01±5.43E+00	**1.22E−01±5.17E−02**
F8*, *l*=50	7.76E+01±2.15E−01	3.12E+01±5.24E−01	4.56E+02±2.74E+00	6.80E+01±1.28E−01	**6.34E−01±2.58E−02**

Here [a±b] indicates the mean and corresponding standard deviations of the error values. F denotes a noiseless benchmark function, F* denotes a noisy benchmark function, and *l* denotes the number of fitness re-samples. The best performance is in bold font in each row. DE, differential evolution; SaDE, self-adaptive differential evolution; PSO, particle swarm optimization; CPSO, particle swarm optimization with constriction; BBO, biogeography-based optimization.

Table 3. Simulation results for F9–F16.

Function	DE	SaDE	PSO	CPSO	BBO
F9	8.97E+00±1.54E−01	**0.00E+00±0.00E+00**	7.24E+01±4.19E+00	4.46E−08±3.29E−09	8.47E−16±2.93E−17
F9*, l=1	1.16E−01±2.47E−02	**3.47E−02±2.56E−03**	6.71E+01±5.36E+00	8.98E−02±4.10E−03	7.00E+00±1.42E−01
F9*, l=5	6.35E+00±7.01E−01	**4.30E−06±5.44E−07**	2.58E+01±1.29E+00	2.45E−03±6.79E−04	9.61E−01±1.25E−02
F9*, l=20	8.42E+00±7.54E−01	**2.19E−04±6.36E−05**	736E+01±2.87E+00	5.38E−02±2.20E−03	7.38E+00±1.45E−01
F9*, l=50	8.61E+00±9.62E−01	**5.28E−04±7.17E−05**	4.55E+01±3.26E+00	9.09E−02±1.12E−03	8.64E+00±9.40E−01
F10	8.21E+00±7.49E−01	**1.16E+00±4.78E−01**	2.34E+01±7.70E+00	8.95E+01±4.36E+00	1.23E+00±3.40E−01
F10*, l=1	3.31E+00±8.25E−01	1.96E+00±3.22E−01	8.49E+01±9.14E+00	**1.16E+00±5.09E−01**	8.52E+00±7.74E−01
F10*, l=5	4.19E+00±3.28E−01	**1.08E+00±1.07E−01**	5.56E+01±8.12E+00	8.99E+00±2.31E−01	1.17E+00±3.04E−01
F10*, l=20	7.80E+00±4.45E−01	**1.06E+00±5.51E−01**	8.53E+01±7.70E+00	6.77E+00±1.49E−01	1.15E+00±7.36E−01
F10*, l=50	2.11E+00±5.32E−01	**1.39E+00±4.37E−01**	8.41E+01±9.55E+00	5.42E+00±7.80E−01	1.56E+00±6.27E−01
F11	4.07E+01±1.37E+00	7.55E+00±2.10E−01	5.66E+02±6.18E+01	5.11E+01±3.46E+00	**1.57E+00±9.88E−01**
F11*, l=1	1.57E+00±4.67E−01	**1.15E+00±4.12E−01**	1.89E+01±1.19E+00	6.48E+00±7.19E−01	1.59E+01±7.04E+00
F11*, l=5	1.37E+02±1.85E+01	1.05E+02±5.64E+01	1.91E+02±8.04E+01	6.30E+02±1.22E+01	**6.47E+01±4.72E+00**
F11*, l=20	1.96E+03±6.03E+02	3.16E+03±1.77E+02	1.46E+04±9.19E+02	4.51E+04±2.45E+03	**2.36E+02±3.19E+01**
F11*, l=50	2.11E+03±8.83E+02	4.35E+03±3.47E+02	2.29E+04±3.48E+02	3.71E+04±4.39E+03	**8.45E+02±3.22E+01**
F12	7.26E+02±3.80E+00	**5.68E−03±2.19E−04**	3.84E+03±4.65E+02	7.14E−01±3.28E−02	9.13E−03±7.55E−04
F12*, l=1	4.52E+02±2.11E+01	7.86E+02±5.39E+01	7.91E+04±2.44E+02	**1.19E+00±7.04E−01**	8.62E+07±3.12E+06
F12*, l=5	3.72E+03±4.36E+02	8.47E+03±7.58E+02	8.26E+05±7.61E+03	7.00E+03±2.36E+02	**4.25E+03±7.89E+02**
F12*, l=20	5.88E+03±1.96E+01	4.32E+00±6.33E−01	8.90E+05±4.42E+03	6.02E+03±5.11E+01	**2.16E−01±3.28E−02**
F12*, l=50	4.45E+04±1.22E+03	1.06E+00±3.50E−01	3.74E+06±1.56E+04	7.42E+02±4.36E+01	**4.45E−01±2.35E−02**
F13	9.65E+01±7.47E+00	**7.84E−03±7.13E−04**	2.35E+00±4.17E−01	8.47E−01±9.00E−02	1.23E−02±5.31E−02
F13*, l=1	3.47E+01±5.16E+00	4.50E+01±7.64E+00	9.02E+00±1.15E−01	1.29E+00±7.02E−01	**2.87E−02±8.09E−03**
F13*, l=5	7.78E+01±5.84E+00	**6.32E−03±1.40E−03**	8.94E+00±6.30E−01	7.36E−01±2.37E−02	4.25E−02±3.78E−03
F13*, l=20	2.36E+01±8.96E+00	**4.55E−02±2.39E−03**	7.98E+00±1.66E−01	2.48E−01±5.51E−02	8.91E−01±1.59E−02
F13*, l=50	3.55E+01±7.89E+00	**7.41E−02±2.35E−03**	7.21E+00±4.38E−01	3.19E−01±4.26E−02	9.68E−01±3.47E−02
F14	8.96E+00±2.47E−01	7.64E−01±6.38E−02	9.74E+00±3.25E−01	**2.46E−01±7.55E−02**	7.85E+00±9.62E−01
F14*, l=1	7.89E+00±5.34E−01	**2.01E+00±8.95E−01**	4.26E+00±8.51E−01	3.59E+00±6.30E−01	3.70E+00±4.66E−01
F14*, l=5	2.35E+00±7.89E−01	2.04E+00±7.30E−01	3.22E+00±7.84E−01	**1.11E+00±4.18E−01**	6.42E+00±1.19E−01
F14*, l=20	3.24E+00±8.74E−01	6.42E+00±7.19E−01	1.25E+00±7.99E−01	**1.09E+00±7.69E−01**	8.45E+00±2.36E−01
F14*, l=50	9.32E+00±2.58E−01	5.02E+00±4.77E−01	6.14E+00±2.38E−01	**1.26E+00±3.01E−01**	7.80E+00±5.62E−01
F15	8.17E+02±9.30E+01	9.17E+02±9.30E+01	7.65E+02±4.16E+01	**1.01E+01±4.32E+00**	7.54E+01±3.25E+00
F15*, l=1	1.83E+02±3.00E+01	1.36E+02±7.18E+01	2.36E+02±8.94E+01	**3.28E+01±3.00E+00**	7.82E+02±1.66E+01
F15*, l=5	5.45E+02±2.74E+01	7.43E+02±8.04E+01	1.23E+02±6.37E+01	**2.66E+01±2.74E+00**	6.47E+01±8.19E+00
F15*, l=20	9.65E+02±7.53E+01	9.06E+02±7.44E+01	7.89E+02±2.30E+01	**1.32E+01±7.53E+00**	2.30E+01±1.74E+00
F15*, l=50	9.62E+02±1.66E+01	1.12E+02±4.52E+01	1.19E+02±5.99E+01	**1.62E+01±1.66E+00**	5.60E+01±3.05E+00
F16	1.12E+01±5.29E+00	**7.71E+00±1.12E+00**	4.41E+02±8.54E+01	4.41E+02±3.22E+01	9.60E+02±1.15E+01
F16*, l=1	4.37E+01±2.16E+00	**2.34E+00±4.50E−01**	7.53E+02±3.26E+01	3.58E+02±7.36E+01	4.72E+02±3.41E+01
F16*, l=5	3.35E+01±7.36E+00	**1.06E+01±3.16E+00**	9.70E+02±4.15E+01	2.13E+02±1.18E+01	2.63E+02±1.74E+01
F16*, l=20	5.11E+01±2.89E+00	**1.15E+01±7.65E+00**	7.88E+02±7.23E+01	4.39E+02±2.94E+01	9.12E+02±7.65E+01
F16*, l=50	7.87E+01±1.23E+00	**1.36E+01±4.22E+00**	9.61E+02±4.38E+01	2.74E+02±5.20E+01	7.30E+02±5.99E+01

Abbreviations as in Table 2.

Table 4. Simulation results for F17–F25.

Function	DE	SaDE	PSO	CPSO	BBO
F17	9.62E+01±1.17E+00	**1.29E+01±4.50E+00**	7.98E+02±2.64E+01	3.38E+01±7.22E+00	7.34E+02±2.19E+01
F17*, l=1	7.86E+02±4.35E+01	**3.14E+01±5.92E+00**	3.27E+03±5.11E+02	7.35E+02±2.17E+01	5.36E+03±4.28E+02
F17*, l=5	7.01E+01±5.64E+00	**1.32E+01±4.26E+00**	7.65E+02±7.61E+01	2.74E+02±1.24E+01	2.18E+02±7.60E+01
F17*, l=20	3.22E+01±1.20E+00	**2.71E+01±5.39E+00**	2.01E+02±1.99E+01	3.47E+02±7.60E+01	5.34E+02±4.78E+01
F17*, l=50	7.16E+01±4.42E+00	**2.65E+01±1.98E+00**	8.33E+02±4.25E+01	5.52E+02±4.31E+01	9.07E+02±3.26E+01
F18	9.28E+02±4.36E+01	**7.65E+00±2.28E+00**	4.21E+01±7.65E+00	8.96E+00±1.16E+00	7.36E+02±9.64E+01
F18*, l=1	5.44E+03±3.17E+02	2.47E+03±5.44E+02	**9.63E+02±7.14E+01**	2.74E+03±1.29E+02	5.28E+03±7.78E+02
F18*, l=5	7.89E+02±4.25E+01	**9.19E+00±6.39E+00**	2.35E+01±4.42E+00	3.96E+02±7.14E+01	3.45E+02±4.11E+01
F18*, l=20	7.03E+03±5.54E+02	**7.36E+01±2.17E+00**	3.27E+02±1.28E+01	6.33E+03±4.10E+02	1.29E+03±3.74E+02
F18*, l=50	2.26E+02±4.11E+01	**1.33E+01±1.12E+00**	5.34E+01±6.62E+00	9.64E+02±7.35E+01	5.55E+02±4.31E+01
F19	7.69E+02±7.41E+01	3.87E+02±1.04E+01	4.25E+02±3.63E+01	8.50E+02±7.41E+01	**9.90E+01±4.47E+00**
F19*, l=1	9.34E+02±5.21E+01	**4.22E+02±4.45E+01**	1.17E+02±4.58E+01	7.41E+02±5.21E+01	2.79E+02±7.28E+01
F19*, l=5	4.49E+02±1.24E+01	4.29E+02±1.07E+01	9.64E+02±5.47E+01	9.64E+02±9.63E+01	**1.86E+02±7.42E+01**
F19*, l=20	2.26E+02±7.62E+01	7.86E+02±3.78E+01	1.25E+02±4.81E+01	7.32E+02±1.70E+01	**7.70E+01±5.38E+00**
F19*, l=50	7.65E+02±7.86E+01	5.39E+02±1.12E+01	2.89E+02±5.62E+01	6.30E+02±4.14E+01	**9.62E+01±3.26E+00**
F20	4.41E+02±3.76E+01	**2.35E+01±5.39E+00**	8.62E+02±1.19E+01	9.98E+01±4.10E+00	8.27E+01±3.19E+00
F20*, l=1	8.94E+02±4.11E+01	**4.11E+01±7.48E+00**	6.58E+03±4.27E+02	9.78E+02±2.33E+01	7.60E+02±9.64E+01
F20*, l=5	9.68E+02±4.49E+01	**2.57E+01±2.45E+00**	7.35E+03±3.25E+02	7.24E+02±7.56E+01	1.17E+02±2.64E+01
F20*, l=20	3.25E+02±1.08E+01	**7.72E+01±7.39E+00**	7.89E+03±2.36E+02	8.29E+02±7.84E+01	9.60E+02±8.85E+01
F20*, l=50	5.53E+02±2.26E+01	**4.63E+01±6.34E+00**	1.80E+03±7.98E+02	9.66E+02±2.28E+01	7.73E+02±1.22E+01
F21	7.65E+02±4.33E+01	8.17E+02±5.64E+01	7.01E+02±8.65E+01	**1.55E+02±2.64E+01**	9.60E+02±1.77E+01
F21*, l=1	9.07E+02±3.39E+01	5.36E+02±1.98E+01	**4.34E+02±2.29E+01**	7.50E+02±7.45E+01	8.59E+02±3.21E+01
F21*, l=5	4.18E+02±7.45E+01	4.38E+02±7.24E+01	7.85E+02±1.99E+01	**7.89E+01±6.30E+00**	7.89E+02±2.37E+01
F21*, l=20	8.67E+02±1.12E+01	7.74E+02±5.59E+01	7.14E+02±3.26E+01	**1.14E+01±9.74E+00**	4.45E+02±7.85E+01
F21*, l=50	7.05E+02±7.82E+01	1.15E+02±6.63E+01	4.78E+02±3.64E+01	**2.30E+01±5.48E+00**	1.04E+02±4.33E+01
F22	1.23E+01±8.96E+00	**8.66E+00±4.25E+00**	9.25E+01±7.84E+00	8.82E+01±3.28E+00	9.25E+00±3.64E+00
F22*, l=1	3.25E+01±9.45E+00	**7.17E+00±2.34E+00**	7.84E+01±4.56E+00	4.79E+01±5.16E+00	7.89E+01±2.17E+00
F22*, l=5	4.18E+01±3.25E+00	**1.54E+01±7.89E+00**	3.26E+01±2.26E+00	2.35E+01±7.75E+00	4.56E+01±6.34E+00
F22*, l=20	7.65E+01±4.72E+00	**5.63E+01±1.16E+00**	7.89E+01±5.96E+00	9.00E+01±1.12E+00	7.82E+01±1.15E+00
F22*, l=50	1.39E+01±1.17E+00	**1.14E+01±5.31E+00**	5.56E+01±7.21E+00	7.48E+01±4.35E+00	6.32E+01±4.36E+00
F23	7.78E+02±7.84E+01	1.86E+02±7.14E+01	7.84E+02±7.45E+01	8.86E+02±5.30E+01	**9.63E+01±4.45E+00**
F23*, l=1	1.25E+02±1.18E+01	**1.05E+02±4.29E+01**	9.36E+03±8.12E+02	2.34E+02±1.99E+01	2.39E+02±7.78E+01
F23*, l=5	4.26E+03±2.47E+02	3.16E+03±6.63E+02	7.12E+03±7.89E+02	3.48E+03±3.28E+02	**2.05E+02±4.73E+01**
F23*, l=20	5.74E+03±5.46E+02	7.45E+02±1.15E+01	7.46E+03±9.21E+02	7.79E+02±9.65E+01	**8.25E+01±3.14E+00**
F23*, l=50	3.25E+03±3.28E+02	6.33E+02±3.17E+01	2.36E+03±7.48E+02	6.25E+02±5.17E+01	**1.29E+01±3.07E+00**
F24	9.65E+02±3.28E+01	**1.36E+02±7.14E+00**	8.56E+02±3.21E+01	3.16E+02±2.44E+01	8.24E+03±4.31E+02
F24*, l=1	4.17E+02±7.85E+01	5.42E+02±3.16E+01	9.19E+02±5.79E+01	**1.55E+02±1.16E+01**	7.89E+03±2.36E+02
F24*, l=5	2.36E+02±2.36E+01	**2.17E+02±7.70E+01**	3.25E+02±6.55E+01	9.36E+02±4.28E+01	4.57E+03±1.44E+02
F24*, l=20	5.44E+02±4.25E+01	**3.29E+02±2.29E+01**	4.28E+02±7.13E+01	7.14E+02±7.77E+01	8.96E+03±8.65E+02
F24*, l=50	1.27E+02±3.27E+01	**1.03E+02±8.33E+01**	6.44E+02±6.21E+01	9.63E+02±9.64E+01	1.12E+03±4.23E+02
F25	9.60E+02±5.32E+01	**7.89E+01±7.41E+00**	9.60E+02±9.57E+01	7.19E+02±6.69E+01	6.32E+02±1.28E+01
F25*, l=1	7.84E+02±1.12E+01	**2.51E+02±2.35E+01**	1.36E+03±3.28E+01	6.45E+02±2.57E+01	4.37E+02±5.36E+01
F25*, l=5	4.58E+03±2.37E+02	**3.74E+02±7.46E+01**	7.89E+03±6.21E+02	7.36E+02±1.14E+01	7.58E+02±9.62E+01
F25*, l=20	9.65E+03±4.25E+02	**1.22E+02±9.03E+01**	5.24E+03±9.85E+02	8.61E+02±6.78E+01	1.74E+02±3.51E+01
F25*, l=50	3.87E+03±1.46E+02	**4.58E+02±7.31E+01**	4.11E+03±2.26E+02	4.64E+02±9.15E+01	9.44E+02±3.28E+01

Abbreviations as in Table 2.

Table 5. Wilcoxon test results.

Function	BBO vs. DE	BBO vs. SaDE	BBO vs. PSO	BBO vs. CPSO	Function	BBO vs. DE	BBO vs. SaDE	BBO vs. PSO	BBO vs. CPSO
F1	X-o	X-o	X-o	X-o	F14	–	o-X	–	o-X
F1*, *l*=1	o-X	o-X	X-o	o-X	F14*, *l*=1	–	–	–	–
F1*, *l*=5	X-o	X-o	X-o	X-o	F14*, *l*=5	–	–	–	–
F1*, *l*=20	X-o	X-o	X-o	X-o	F14*, *l*=20	–	–	–	–
F1*, *l*=50	X-o	X-o	X-o	X-o	F14*, *l*=50	–	–	–	–
F2	o-X	o-X	X-o	X-o	F15	X-o	X-o	X-o	–
F2*, *l*=1	o-X	o-X	X-o	–	F15*, *l*=1	–	–	–	o-X
F2*, *l*=5	–	o-X	X-o	–	F15*, *l*=5	X-o	X-o	X-o	–
F2*, *l*=20	–	o-X	X-o	–	F15*, *l*=20	X-o	X-o	X-o	–
F2*, *l*=50	–	o-X	X-o	–	F15*, *l*=50	X-o	X-o	X-o	–
F3	o-X	o-X	X-o	o-X	F16	o-X	o-X	–	–
F3*, *l*=1	o-X	o-X	–	o-X	F16*, *l*=1	o-X	o-X	–	–
F3*, *l*=5	–	o-X	–	–	F16*, *l*=5	o-X	o-X	–	–
F3*, *l*=20	–	o-X	–	o-X	F16*, *l*=20	o-X	o-X	–	–
F3*, *l*=50	–	o-X	–	o-X	F16*, *l*=50	o-X	o-X	–	–
F4	–	–	X-o	–	F17	o-X	o-X	–	o-X
F4*, *l*=1	o-X	o-X	–	o-X	F17*, *l*=1	o-X	o-X	–	o-X
F4*, *l*=5	–	o-X	–	o-X	F17*, *l*=5	o-X	o-X	–	–
F4*, *l*=20	–	–	–	o-X	F17*, *l*=20	o-X	o-X	–	–
F4*, *l*=50	–	–	–	o-X	F17*, *l*=50	o-X	o-X	–	–
F5	o-X	o-X	X-o	X-o	F18	–	o-X	o-X	o-X
F5*, *l*=1	o-X	o-X	–	o-X	F18*, *l*=1	–	–	o-X	–
F5*, *l*=5	–	o-X	X-o	–	F18*, *l*=5	–	o-X	o-X	–
F5*, *l*=20	–	o-X	X-o	–	F18*, *l*=20	–	o-X	o-X	–
F5*, *l*=50	–	o-X	–	o-X	F18*, *l*=50	–	o-X	o-X	–
F6	X-o	X-o	X-o	X-o	F19	X-o	X-o	X-o	X-o
F6*, *l*=1	o-X	o-X	–	o-X	F19*, *l*=1	–	–	–	–
F6*, *l*=5	–	X-o	X-o	X-o	F19*, *l*=5	–	–	–	–
F6*, *l*=20	–	X-o	X-o	X-o	F19*, *l*=20	X-o	X-o	X-o	X-o
F6*, *l*=50	X-o	X-o	X-o	X-o	F19*, *l*=50	X-o	X-o	X-o	X-o
F7	X-o	X-o	–	X-o	F20	X-o	–	X-o	–
F7*, *l*=1	–	o-X	o-X	o-X	F20*, *l*=1	–	o-X	X-o	–
F7*, *l*=5	–	o-X	o-X	o-X	F20*, *l*=5	–	o-X	X-o	–
F7*, *l*=20	–	o-X	o-X	o-X	F20*, *l*=20	–	o-X	X-o	–
F7*, *l*=50	–	o-X	o-X	o-X	F20*, *l*=50	–	o-X	X-o	–
F8	X-o	–	X-o	o-X	F21	–	–	–	–
F8*, *l*=1	–	X-o	X-o	X-o	F21*, *l*=1	–	–	–	–
F8*, *l*=5	–	X-o	X-o	X-o	F21*, *l*=5	–	–	–	o-X
F8*, *l*=20	X-o	X-o	X-o	X-o	F21*, *l*=20	–	–	–	o-X
F8*, *l*=50	X-o	X-o	X-o	X-o	F21*, *l*=50	–	–	–	o-X
F9	X-o	o-X	X-o	X-o	F22	X-o	–	X-o	X-o
F9*, *l*=1	–	o-X	–	o-X	F22*, *l*=1	–	o-X	–	–
F9*, *l*=5	–	o-X	X-o	o-X	F22*, *l*=5	–	–	–	–
F9*, *l*=20	–	o-X	–	o-X	F22*, *l*=20	–	–	–	–
F9*, *l*=50	–	o-X	–	o-X	F22*, *l*=50	–	–	–	–
F10	–	–	–	X-o	F23	X-o	X-o	X-o	X-o
F10*, *l*=1	–	–	–	–	F23*, *l*=1	–	–	X-o	–
F10*, *l*=5	–	–	–	–	F23*, *l*=5	–	X-o	–	X-o
F10*, *l*=20	–	–	–	–	F23*, *l*=20	X-o	X-o	X-o	X-o
F10*, *l*=50	–	–	–	–	F23*, *l*=50	X-o	X-o	X-o	X-o
F11	X-o	–	X-o	X-o	F24	o-X	o-X	o-X	o-X
F11*, *l*=1	o-X	o-X	–	–	F24*, *l*=1	o-X	o-X	o-X	o-X
F11*, *l*=5	–	X-o	X-o	X-o	F24*, *l*=5	o-X	o-X	o-X	o-X
F11*, *l*=20	X-o	X-o	X-o	X-o	F24*, *l*=20	o-X	o-X	o-X	o-X
F11*, *l*=50	X-o	X-o	X-o	X-o	F24*, *l*=50	o-X	o-X	o-X	o-X
F12	X-o	–	X-o	X-o	F25	–	o-X	–	–
F12*, *l*=1	o-X	–	X-o	o-X	F25*, *l*=1	–	–	X-o	–
F12*, *l*=5	–	X-o	X-o	–	F25*, *l*=5	X-o	–	X-o	–
F12*, *l*=20	X-o	X-o	X-o	X-o	F25*, *l*=20	X-o	–	X-o	–

(continued)

Table 5. Continued

Function	BBO vs. DE	BBO vs. SaDE	BBO vs. PSO	BBO vs. CPSO	Function	BBO vs. DE	BBO vs. SaDE	BBO vs. PSO	BBO vs. CPSO
F12*, l=50	X-o	X-o	X-o	X-o	F25*, l=50	X-o	–	X-o	–
F13	X-o	o-X	X-o	X-o	**Total**	**37, 26**	**31, 59**	**61, 14**	**35, 35**
F13*, l=1	X-o	X-o	X-o	X-o					
F13*, l=5	X-o	o-X	X-o	X-o					
F13*, l=20	X-o	o-X	X-o	–					
F13*, l=50	X-o	o-X	X-o	–					

Abbreviations as in Table 2. 'X-o' shows that the left algorithm is significantly better than the right one, and 'o-X' shows that the right algorithm is significantly better than the left one. The 'Total' row at the end of the table shows the number of times BBO outperforms DE, SaDE, PSO, CPSO, and vice versa.

- There are 70 statistically significant differences between CPSO and BBO, including:
  - 35 groups of data for which BBO is better;
  - 35 groups of data for which CPSO is better.

The statistical results show that for the noiseless and noisy benchmark functions, SaDE performs best, BBO and CPSO perform second best, DE performs fourth best, and PSO performs worst. The superior performance of SaDE is apparently due to its self-adaptive nature. This implies that a similar self-adaptation strategy could also significantly improve performance in PSO and BBO.

Comparisons with Kalman filter-based BBO (KBBO)

To illustrate the performance of BBO further combined with re-sampling, we compare it with Kalman filter-based BBO (Du, 2009), which is called KBBO. We use $l = 5$ fitness re-samples for BBO with re-sampling because it offers good performance and uses a relatively small number of fitness samples. The parameters used in these two BBO algorithms are the same as those described in the previous section. The number of total fitness evaluations is 100,000 and all functions are optimized over 25 independent runs. We use the same set of initial random populations to evaluate these two BBO algorithms. The results of solving the 25 noisy benchmark functions are given in Table 6.

Table 6 shows that for noisy benchmark functions, BBO combined with re-sampling performs the best on 12 functions, and KBBO performs the best on the other 13 functions. This result shows that BBO with re-sampling achieves almost the same performance as KBBO for noisy optimization problems. The average computational times of these two BBO algorithms are shown in the last row of Table 6. We see that the average computational time of BBO with re-sampling is much lower than that of KBBO.

Table 7 shows the results of Wilcoxon tests between BBO with re-sampling, and KBBO. Out of 25 groups of data, we find that there are 19 statistically significant differences between the two algorithms, including nine groups of data for which BBO is better and 10 groups of data for which KBBO is better.

Table 6. Comparisons of simulation results for biogeography-based optimization (BBO) with the number of fitness re-samples l=5, and BBO using the Kalman filter (KBBO).

Function	KBBO	Re-sampled BBO with l=5
F1	5.27E−02±1.15E−03	**7.48E−07±2.26E−08**
F2	**9.54E−05±5.56E−06**	9.34E−02±7.25E−03
F3	**3.27E−03±4.36E−04**	7.52E+01±3.39E+00
F4	**9.00E−01±1.25E−02**	4.58E+00±7.86E−01
F5	7.58E+02±3.24E+01	**1.23E+01±4.11E+00**
F6	**9.46E−06±3.15E−07**	8.27E−06±1.25E−07
F7	4.26E−01±6.32E−02	**5.36E−03±7.83E−04**
F8	1.74E+02±3.28E+01	**1.02E+00±2.38E−01**
F9	**9.64E−04±1.19E−05**	9.65E−01±7.14E−02
F10	**5.63E−02±2.58E−03**	8.03E+00±1.24E−01
F11	7.81E−02±4.56E−03	**9.07E−02±5.44E−03**
F12	3.38E+03±7.24E+02	**7.58E+03±3.16E+02**
F13	**8.63E−04±2.87E−05**	7.78E−02±5.36E−03
F14	**4.27E−01±6.35E−02**	3.02E+00±5.11E−01
F15	9.36E+01±2.40E+00	**9.25E+01±2.78E+00**
F16	**7.89E+02±5.59E+01**	2.14E+02±4.37E+01
F17	3.27E+02±1.24E+01	**6.47E+02±5.32E+01**
F18	7.89E+02±4.28E+01	**8.21E+02±6.35E+01**
F19	**9.00E+01±4.15E+00**	8.34E+01±7.78E+00
F20	**7.23E+02±5.31E+01**	3.20E+02±8.60E+01
F21	**7.80E+02±4.51E+01**	7.64E+02±5.32E+01
F22	3.25E+01±1.17E+00	**7.78E+1±4.28E+00**
F23	7.65E+01±4.32E+00	**9.61E+01±5.33E+00**
F24	6.10E+03±9.22E+02	**1.02E+03±7.20E+02**
F25	**4.25E+02±4.10E+01**	6.34E+02±9.60E+01
CPU time	6.22	4.53

Here [a±b] indicates the mean and corresponding standard deviations of the error values. The last row shows the average computational time in minutes. The best performance is in bold font in each row.

We make two general conclusions from these results. First, both BBO combined with re-sampling, and KBBO, alleviate the effects of noise for the benchmark functions that we studied, but a Kalman filter is an optimal estimator for the states of a linear dynamic system, so KBBO may be a better method if the model of the noise's effect on the fitness values is well known. However, the Kalman filter requires multiple tuning parameters (Simon, 2006), and so KBBO may be difficult to tune. A poorly tuned Kalman filter may give misleading

Table 7. Wilcoxon test results of biogeography-based optimization (BBO) with re-sampling, and BBO using the Kalman filter (KBBO).

Function	KBBO vs. BBO (*l*=5)	Function	KBBO vs. BBO (*l*=5)	Function	KBBO vs. BBO (*l*=5)
F1	o-X	F10	X-o	F18	–
F2	X-o	F11	o-X	F19	X-o
F3	X-o	F12	o-X	F20	–
F4	X-o	F13	X-o	F21	X-o
F5	o-X	F14	X-o	F22	–
F6	–	F15	–	F23	o-X
F7	o-X	F16	–	F24	o-X
F8	o-X	F17	o-X	F25	X-o
F9	X-o	–	–	**Total**	**10, 9**

If the difference between the algorithms is significant with a level of significance or less, the pairs are marked as follows: X-o shows that the left algorithm is significantly better than the right one; and o-X shows that the right algorithm is significantly better than the left one. The 'Total' row at the end of the table shows that KBBO outperforms BBO by a score of 10 to 9.

estimation results. Second, BBO combined with re-sampling is faster than KBBO due to the computational complexity of the Kalman filter.

Conclusion

We investigated the effect of fitness function noise on BBO performance using a Markov model, and we used re-sampling to alleviate the effect of random noise on the fitness function evaluations of numerical benchmark functions. Analysis showed the amount by which migration between candidate solutions, which is the most critical operation of BBO, is corrupted by fitness function evaluation noise. Analysis also showed that the effect of noise is alleviated by high mutation rates, although high mutation rates might themselves be detrimental to BBO performance. The analysis was confirmed with an example using a BBO Markov model.

We used re-sampling in BBO and other EAs to deal with random noise. We also compared BBO with re-sampling, and BBO augmented with Kalman filtering. Our numerical simulations showed the following: 1) BBO is a powerful EA for noiseless benchmark functions, but fitness function evaluation noise is indeed a problem for BBO; 2) SaDE performs best on noisy optimization problems, BBO and CPSO perform second best, DE performs third best, and PSO performs worst; 3) BBO with re-sampling achieves the same optimization performance as KBBO, but uses less computational time; 4) although re-sampling is simple, it can greatly improve the performance of BBOandotherEAsinnoisyenvironments.

This paper focused on the fitness of candidate solutions contaminated by additive, normally distributed noise. For future work, there are several important directions. First, in many real-world applications, different types of fitness function noise can be encountered, so it is of interest to combine BBO with re-sampling to address other types of fitness function noise. Furthermore, other types of noise problems (besides fitness function noise) can arise in optimization. For example, in distributed optimization, some nodes might temporarily drop out of the algorithm due to communication glitches; or during experimental optimization, some parameters might be corrupted during fitness function evaluation. Future research could explore the effects of these and other types of noise on EA performance.

The second important direction for future work is to explore the optimization performance of BBO combined with other noise-handling methods, e.g. dynamic re-sampling, which uses different re-sampling rates at different points in the search domain. The third important direction for future work is to investigate the optimization ability of other BBO variations on noisy problems. The fourth direction for future work is to develop hybrid BBO algorithms for noisy problems (i.e. BBO combined with other optimization algorithms).

Acknowledgements

The comments of the anonymous reviewers were helpful in improving this paper from its original version.

Conflict of interest

The authors declare that there is no conflict of interest.

Funding

This material is based upon work supported by the National Science Foundation under grant number 0826124, the National Natural Science Foundation of China under grant numbers 61305078, 61074032 and 61179041 and the Shaoxing City Public Technology Applied Research Project under grant number 2013B70004.

References

Al-Rifaie MM and Blackwell T (2012) Bare bones particle swarms with jumps. *Lecture Notes in Computer Science* 7461: 49–60.

Arnold DV and Beyer HG (2003) On the effects of outliers on evolutionary optimization. *Intelligent Data Engineering and Automated Learning*. Berlin: Springer, pp. 151–160.

Beyer HG and Sendhoff B (2007) Robust optimization – a comprehensive survey. *Computer Methods in Applied Mechanics and Engineering* 196(33): 3190–3218

Bhattacharya A and Chattopadhyay P (2010) Biogeography-based optimization for different economic load dispatch problems. *IEEE Transactions on Power Systems* 25(2): 1064–1077.

Bratton D and Kennedy J (2007) Defining a standard for particle swarm optimization. In:*Proceedings of IEEE Swarm Intelligence Symposium*, Honolulu, HI, pp. 120–127.

Chen T, He J, Chen G, et al. (2010a) Choosing selection pressure for wide-gap problems. *Theoretical Computer Science* 411(6): 926–934.

Chen T, Tang K, Chen G, et al. (2010b) Analysis of computational time of simple estimation of distribution algorithms. *IEEE Transactions on Evolutionary Computation* 14(1): 1–22.

Clerc M (2006) *Particle Swarm Optimization*. London: ISTE Publishing.

Clerc M and Kennedy J (2002) The particle swarm – Explosion, stability and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation* 6(1): 58–73.

Das S and Suganthan PN (2011) Differential evolution – a survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation* 15(1): 4–31.

Das S, Konar A and Chakraborty U. K (2005) Improved differential evolution algorithms for handling noisy optimization problems. In:*Proceedings of IEEE Congress on Evolutionary Computation*, Edinburgh, UK, pp. 1691–1698.

Demsar J (2006) Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research* 7: 1–30.

Derrac J, Garcia S, Molina D, et al. (2011) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation* 1(1): 3–18.

Dong W, Chen T, Tino P, et al. (2013) Scaling up estimation of distribution algorithms for continuous optimization. *IEEE Transactions on Evolutionary Computation* 17(6), 797–822.

Du D (2009) *Biogeography-based optimization: synergies with evolutionary strategies, immigration refusal, and Kalman filters*. Master's thesis, Cleveland State University, Cleveland, OH.

Du D, Simon D and Ergezer M (2009) Biogeography-based optimization combined with evolutionary strategy and immigration refusal. In: *Proceedings of the IEEE Conference on Systems, Man, and Cybernetics*, San Antonio, TX, pp. 1023–1028.

Eberhart R and Shi Y (2004) Special issue on particle swarm optimization. *IEEE Transactions on Evolutionary Computation* 3(4): 201–228.

Eberhart R, Shi Y and Kennedy J (2001) *Swarm Intelligence*. San Mateo, CA: Morgan Kaufmann.

Eberhart RC and Shi Y (2000) Comparing inertia weights and constriction factors in particle swarm optimization. *IEEE Congress on Evolutionary Computation*, pp. 84–88.

Elton C (1958) *Ecology of Invasions by Animals and Plants*. London: Chapman & Hall.

Ergezer M, Simon D and Du D (2009) Oppositional biogeography-based optimization. In:*Proceedings of the IEEE Conference on Systems, Man, and Cybernetics*, San Antonio, TX, pp. 1035–1040.

Fitzpatrick JM and Grefenstette JJ (1988) Genetic algorithms in noisy environments. *Machine Learning* 3(2): 101–120.

Hansen N, Niederberger A, Guzzella L, et al. (2009) A method for handling uncertainty in evolutionary optimization with an application to feedback control of combustion. *IEEE Transactions on Evolutionary Computation* 13(1): 180–197.

Harding S (2006) *Animate Earth*. White River Junction, VT: Chelsea Green Publishing Company.

Jin Y and Branke J (2005) Evolutionary optimization in uncertain environments – a survey. *IEEE Transactions on Evolutionary Computation* 9(3): 303–317.

Keel L and Bhattacharyya S (1997) Robust, fragile, or optimal?*IEEE Transactions on Automatic Control* 42(8): 1098–1105.

Kheng CW, Chong SY and Lim M (2012) Centroid-based memetic algorithm – adaptive Lamarckian and Baldwinian learning. *International Journal of Systems Science* 43(7): 1193–1216.

Krink T, Filipic B and Fogel G (2004) Noisy optimization problem – a particular challenge for differential evolution? In: *Proceedings of the 2004 Congress on Evolutionary Computation*, Porland, OR, pp. 19–23.

Lenton T (1998) Gaia and natural selection. *Nature* 394(6692): 439–447.

Liu B, Zhang X and Ma H (2008) Hybrid differential evolution for noisy optimization. In:*Proceedings of IEEE Congress on Evolutionary Computation*, Hong Kong, China, pp. 587–592.

Lovelock J (1990) Hands up for the Gaia hypothesis. *Nature* 344(6262): 100–102.

Ma H (2010) An analysis of the equilibrium of migration models for biogeography-based optimization. *Information Sciences* 180(18): 3444–3464.

Ma H and Simon D (2011) Blended biogeography-based optimization for constrained optimization. *Engineering Applications of Artificial Intelligence* 24(3): 517–525.

MacArthur R (1955) Fluctuations of animal populations and a measure of community stability. *Ecology* 36(3): 533–536.

MacArthur R and Wilson E (1963) An equilibrium theory of insular zoogeography. *Evolution* 17(4): 373–387.

MacArthur R and Wilson E (1967) *The Theory of Island Biogeography*. Princeton, NJ: Princeton University Press.

Mendel E, Krohling RA and Campos M (2011) Swarm algorithms with chaotic jumps applied to noisy optimization problems, *Information Sciences* 181(20): 4494–4514.

Mininno E and Neri F (2010) A memetic differential evolution approach in noisy optimization. *Memetic Computing* 2(2): 111–135.

Mühlenbein H and Schlierkamp-Voosen D (1993) Predictive models for the breeder genetic algorithm I: continuous parameter optimization. *Evolutionary Computation* 1(1): 25–49.

Neri F, Garcia XT, Cascella GL, et al. (2008) Surrogate assisted local search in PMSM drive design. *COMPEL: The International Journal for Computation and Mathematics in Electrical and Electronic Engineering* 27(3): 573–592.

Onwubolu G and Babu B (2004) *New Optimization Techniques in Engineering*. Berlin: Springer.

Pan H, Wang L and Liu B (2006) Particle swarm optimization for function optimization in noisy environment. *Applied Mathematics and Computation* 181(2): 908–919.

Panchal V, Singh P, Kaur N, et al. (2009) Biogeography based satellite image classification. *International Journal of Computer Science and Information Security* 6(2): 269–274.

Particle Swarm Central, http://www.particleswarm.info/.

Pietro A, While L and Barone L (2004) Applying evolutionary algorithms to problems with noisy, time-consuming fitness functions. In: *Proceedings of the 2004 Congress on Evolutionary Computation*, Portland, OR, pp. 1254–1261.

Price K and Storn R (1997) Differential evolution. *Dr. Dobb's Journal* 22: 18–20.

Rarick R, Simon D, Villaseca FE, et al. (2009) Biogeography-based optimization and the solution of the power flow problem. In:*Proceedings of the IEEE Conference on Systems, Man, and Cybernetics*, San Antonio, TX, pp. 1029–1034.

Schwefel HP (1993) *Evolution and Optimum Seeking*. New York: John Wiley & Sons.

Simon D (2006) *Optimal State Estimation*. New York: John Wiley & Sons.

Simon D (2008) Biogeography-based optimization. *IEEE Transactions on Evolutionary Computation* 12(6): 702–713.

Simon D (2013) *Evolutionary Optimization Algorithms*. New York: John Wiley & Sons.

Simon D, Ergezer M, Du D, et al. (2011) Markov models for biogeography-based optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 41(1): 299–306.
Storn R (1999) System design by constraint adaptation and differential evolution. *IEEE Transactions on Evolutionary Computation* 3(1): 22–34.
Stroud P (2001) Kalman-extended genetic algorithm for search in nonstationary environments with noisy fitness evaluations. *IEEE Transactions on Evolutionary Computation* 5(1): 66–77.
Suganthan PN, Hansen N, Liang JJ, et al. (2005) Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. Technical Report, Nanyang Technological University, Singapore.
Whittaker R and Bush M (1993) Dispersal and establishment of tropical forest assemblages, Krakatoa, Indonesia. In: Miles J, and Walton D (eds) *Primary Succession on Land*. Oxford: Blackwell Science, pp. 147–160.
Yao X, Liu Y and Lin G (1999) Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation* 3(2): 82–102.
Yu X, Tang K, Chen T, et al. (2008) Empirical analysis of evolutionary algorithms with immigrants schemes for dynamic optimization. *Memetic Computing* 1(1): 3–24.
Zhao SZ, Suganthan PN and Das S (2011) Self-adaptive differential evolution with multi-trajectory search for large-scale optimization. *Soft Computing* 15(11): 2175–2185.