Engineering Applications of Artificial Intelligence 26 (2013) 2397–2407

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

On the equivalences and differences of evolutionary algorithms

Haiping Ma [a,c,*], Dan Simon [b], Minrui Fei [c], Zixiang Chen [a]

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

ARTICLE INFO

Article history:
Received 30 December 2012
Received in revised form
22 March 2013
Accepted 3 May 2013
Available online 31 May 2013

Keywords:
Evolutionary algorithms
Genetic algorithm
Biogeography-based optimization
Differential evolution
Evolution strategy
Particle swarm optimization

ABSTRACT

Evolutionary algorithms (EAs) are fast and robust computation methods for global optimization, and have been widely used in many real-world applications. We first conceptually discuss the equivalences of various popular EAs including genetic algorithm (GA), biogeography-based optimization (BBO), differential evolution (DE), evolution strategy (ES) and particle swarm optimization (PSO). We find that the basic versions of BBO, DE, ES and PSO are equal to the GA with global uniform recombination (GA/GUR) under certain conditions. Then we discuss their differences based on biological motivations and implementation details, and point out that their distinctions enhance the diversity of EA research and applications. To further study the characteristics of various EAs, we compare the basic versions and advanced versions of GA, BBO, DE, ES and PSO to explore their optimization ability on a set of real-world continuous optimization problems. Empirical results show that among the basic versions of the algorithms, BBO performs best on the benchmarks that we studied. Among the advanced versions of the algorithms, DE and ES perform best on the benchmarks that we studied. However, our main conclusion is that the conceptual equivalence of the algorithms is supported by the fact that algorithmic modifications result in very different performance levels.

1. Introduction

Evolutionary algorithms (EAs) (Schwefel, 1995; Whitley, 2001; Yao et al., 1999) such as the genetic algorithm (GA) (Reeves and Rowe, 2003; Vose, 1999), biogeography-based optimization (BBO) (Ma, 2010; Simon, 2008), differential evolution (DE) (Das and Suganthan, 2011; Storn and Price, 1997), evolution strategy (ES) (Arnold and Beyer, 2001; Beyer, 1994) and particle swarm optimization (PSO) (Bratton and Kennedy, 2007; Kennedy and Eberhart, 1995; Kennedy, 1997) have received much attention regarding their potential as global optimization methods in real-world applications. Inspired by natural evolution and survival of the fittest in the biological world, EAs are search methods that are different from traditional optimization techniques, and are based on a collective learning process within a population of candidate solutions. In this paper we often use the shorthand term *solution* to refer to a candidate solution. The population in EAs is usually arbitrarily initialized, and each iteration (also called a generation) evolves toward better and better solution regions by means of randomized processes of selection (which is deterministic in some algorithms), mutation, and recombination (which is omitted in some algorithms). The environment delivers quality information (fitness values for maximization problems, and cost values for minimization problems) about candidate solutions. The solutions with high fitness are selected to reproduce more often than those with lower fitness. All solutions have a small mutation chance to introduce innovation into the population. Each EA works on the principles of a different natural phenomena. GA uses survival of the fittest, BBO uses the migration behavior of species between islands, DE uses vector differences of candidate solutions, ES uses self-adaptive mutation rates, and PSO uses the foraging behavior of birds. But all of these EAs have certain features in common, and probabilistically share information between candidate solutions to improve the solution fitness. This makes them applicable to all kinds of optimization problems.

These EAs have been applied to many engineering optimization problems and have proven effective for solving some specific problems, including unimodal, multimodal, and deceptive optimization, constrained optimization, dynamic optimization, noisy optimization, multi-objective optimization, and so on (Ahn, 2006; Chiong et al., 2012; Oltean, 2007). So they are becoming increasingly popular tools to solve various hard optimization problems. Now many efforts have also been devoted to compare these algorithms to each other. Typically, such comparisons have been based on artificial numerical benchmark problems. Most

* Corresponding author at: Department of Electrical Engineering, Shaoxing University, Shaoxing, Zhejiang, China. Tel.: +86 575 88345673; fax: +86 575 88341530.
E-mail address: Mahp@usx.edu.cn (H. Ma).

0952-1976/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.engappai.2013.05.002

studies are to verify that one algorithm outperforms another on a given set of benchmarks. However, there has not been much comparative study of various EAs and their principles of operation. Therefore, it is interesting to discuss and compare the characteristics of popular EAs from the conceptual and algorithmic aspects.

The aim of this paper is to show the equivalences and differences of various popular EAs, including GA, BBO, DE, ES and PSO in a notional as well as in an experimental way. Because these algorithms bear so many similarities due to their reliance on organic evolution, it is not a surprising fact that these algorithms have equivalences under certain conditions. In this study, we formalize a general description of these algorithms and provide detailed theoretical and empirical comparisons. This paper can provide an appropriate overview of the strong similarities of these algorithms to stimulate further discussions.

There has been much previous work comparing various EAs, including comparisons between genetic algorithms, memetic algorithms, particle swarm optimization, ant colony optimization, and shuffled frog leaping (Elbeltagi et al., 2005); comparisons between genetic algorithms and particle swarm optimization (Eberhart and Shi, 1998); comparisons between genetic algorithms and evolution strategies (Hoffmeister and Bäck, 1990); comparisons between memetic algorithms, tabu search, and ant colony optimization (Merz and Freisleben, 1999); and many others (Gao, 2004; Lai et al.,1998; Settles et al., 2003). Those papers focus mostly on the motivation behind the algorithms and differences in performance on benchmarks or specific applications. In this paper we add to the research literature by providing a more extensive comparison, by focusing on similarities and differences between algorithms, by including the recently developed BBO algorithm in our comparison, and by focusing on differences in performance on a larger and more recent set of benchmarks. The benchmarks we study in this paper are recently-proposed real-world continuous problems from the 2011 IEEE Congress on Evolutionary Computation (Das and Suganthan, 2010).

We note that there are many other popular EAs that we could include in our comparison, including ant colony optimization (ACO) (Dorigo et al., 2002; Dorigo and Gambardella, 1997), artificial immune systems (AIS) (Hofmeyr and Forrest, 2000), and artificial bee colony (ABC) optimization (Karaboga and Basturk, 2007). These algorithms may be similar to the five EAs that we examine in this paper, but their similarity has not yet been examined. We restrict our comparison to five algorithms due to space constraints, and we leave the comparison of other algorithms for future work. We chose the five algorithms that we did because GA and ES are two of the basic and foundational approaches to computer intelligence; DE is a mid-generation addition to the EA family that has proven very successful; PSO takes a fundamentally different approach as a swarm intelligence algorithm; and BBO is a typical late addition to the family of EAs. The five algorithms that we chose thus form a representative set rather than a complete set.

The rest of this paper is organized as follows. Section 2 first gives a brief overview of various basic EAs and analyzes their equivalences, and then discusses both their differences and their unique characteristics. Section 3 presents performance comparisons of the basic and advanced EAs on real-world application benchmarks, and Section 4 gives conclusions and directions for future research.

2. Equivalences and differences of EAs

This section first introduces various basic EAs, including GA, BBO, DE, ES and PSO, and then conceptually analyzes their equivalences under special conditions (Section 2.1). This section then discusses their differences based on biologic motivations and algorithmic details (Section 2.2).

2.1. Equivalences of EAs

2.1.1. Genetic algorithms

GAs are popular evolutionary algorithms which were introduced as a computational analogy of adaptive biological systems. They are modeled on natural selection in evolution. GAs use a set of candidate solutions as a population, and use fitness functions to evaluate these candidate solutions. In the process of evolution, the candidate solutions are improved through selection, mutation and recombination (crossover) operators, and then pass on the candidate solutions with the best fitness to the next generation. A general description of one generation of a simple GA is given in Algorithm I.

Algorithm I. A general description of one generation of a simple GA, which is divided into four steps.

Select the best-fit solutions for reproduction
Breed new solutions through recombination (crossover) and mutation operations
Evaluate the fitness of the new solutions
Retain the most fit solutions for the next generation

The simple GA described in Algorithm I is usually the one applied to most problems presented to a GA. The new solutions are obtained each generation by recombination and mutation. Prior to its mutation an offspring is produced by recombining parent solutions:

$$y_k(s) = \begin{cases} y_a(s) & \text{(A) no recombination} \\ y_a(s) \text{ or } y_b(s) & \text{(B) discrete} \\ y_a(s) + \alpha(y_b(s) - y_a(s)) & \text{(C) intermediate} \end{cases} \tag{1a}$$

$$y_k(s) = \begin{cases} y_{a(s)}(s) \text{ or } y_{b(s)}(s) & \text{(D) global, discrete} \\ y_{a(s)}(s) + \alpha(s)(y_{b(s)}(s) - y_{a(s)}(s)) & \text{(E) global, intermediate} \end{cases} \tag{1b}$$

where α and $\alpha(s)$ are contraction factors between 0 and 1, y is the entire population of candidate solutions, a, b, $a(s)$, and $b(s)$ are parent indices, k is the offspring index, and s is the decision variable index of a candidate solution. For example, y_k denotes the kth offspring, and $y_k(s)$ is the sth decision variable of y_k. For options (A), (B), and (C), parent indices a and b are two randomly-selected indices that are independent of decision variable index s, and contraction factor α is also independent of s. For options (D) and (E), $a(s)$ and $b(s)$ are randomly-selected indices that depend on decision variable index s, and $\alpha(s)$ is a contraction factor that depends on s.

By convention all parents in a population have the different mating probabilities, namely, all parents are determined by fitness-based selection, for example, roulette-wheel selection or tournament selection. In the case of discrete recombination option (B) in (1a), the sth decision variable of the offspring is chosen from either of two parents, which may be interpreted as crossover with a varying number of crossover points. In the case of intermediate recombination option (C) in (1a), the sth decision variable of the offspring is the weighted average of the two parents, and the weighting coefficient is α. In the case of the global recombination options (D) and (E) in (1b), $a(s)$ and $b(s)$ are chosen independently for each decision variable index s, which means that many parents can contribute to a single offspring. This results in a higher mixing of genetic information than in the case of options (B) and (C).

2.1.2. Biogeography-based optimization

BBO was introduced in 2008 (Simon, 2008). It is a relatively new evolutionary optimization algorithm which is inspired by biogeography theory. In BBO, a biogeography habitat denotes a candidate optimization problem solution, and it is comprised of a set of features, which are also called decision variables, and which are similar to genes or alleles in GAs. A set of biogeography habitats is an archipelago of islands, and denotes a population of candidate solutions. The number of islands in the archipelago corresponds to the BBO population size. Habitat suitability index (HSI) in biogeography denotes the fitness of a candidate solution. Like other EAs, each candidate solution in BBO probabilistically shares decision variables with other candidate solutions to improve candidate solution fitness. This sharing process is analogous to migration in biogeography. That is, each candidate solution immigrates decision variables from other candidate solutions based on its immigration rate, and emigrates decision variables to other candidate solutions based on its emigration rate. BBO consists of two main steps: migration (which includes both immigration and emigration), and mutation.

Migration is a probabilistic operator that is intended to improve a candidate solution y_k. For each decision variable of a given candidate solution y_k, the candidate solution's immigration rate λ_k is used to probabilistically decide whether or not to immigrate. If immigration is selected, then the emigrating candidate solution y_j is probabilistically chosen based on its emigration rate μ_j. Migration is written as

$$y_k(s) \leftarrow y_j(s) \tag{2}$$

where s is a decision variable index just as in GAs. In BBO, each candidate solution y_k has its own immigration rate λ_k and emigration rate μ_k. A good candidate solution has a relatively high emigration rate and a low immigration rate, while the converse is true for a poor candidate solution. Here, immigration rate λ_k and emigration rate μ_k are based on particular migration curves, such as the linear migration curves shown in Fig. 1, where the maximum immigration rate and maximum emigration rate are both equal to 1. More possibilities for the shape of the migration curves have been discussed in Ma (2010).

Mutation is a probabilistic operator that randomly modifies a decision variable of a candidate solution. The purpose of mutation is to increase diversity among the population, just as in other EAs. A description of one generation of BBO is given in Algorithm II. Note that Algorithm II includes the use of a temporary population z so that migration completes before the original population y is replaced with the new population z at the end of each generation.

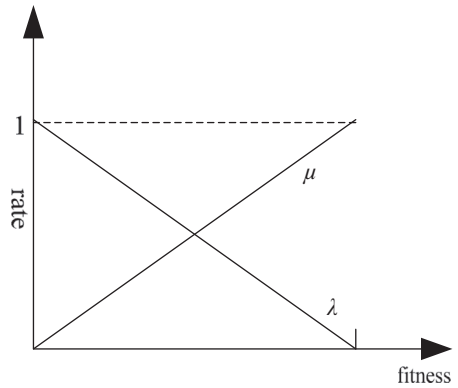

Fig. 1. Linear migration curves for BBO. λ is the immigration rate and μ is the emigration rate, and we assume that the maximum immigration rate and maximum emigration rate are both equal to 1.

Algorithm II. One generation of the BBO algorithm, where N is the population size. y and z are the entire population of candidate solutions, y_k is the kth candidate solution, and $y_k(s)$ is the sth decision variable 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 candidate solution decision variable index s
        Use λ_k to probabilistically decide whether to
  immigrate to z_k
        If immigrating then
            Use {μ} to probabilistically select the emigrating
  solution y_j
            z_k(s)←y_j(s)
        End if
    Next candidate solution decision variable index
    Probabilistically decide whether to mutate z_k
Next solution
y←z
```

The BBO migration strategy is conceptually similar to a combination of two ideas from GAs: global recombination and uniform crossover, which has been explained in detail in Simon et al. (2011), and is reviewed here. As the name suggests in option (D) of (1b), global recombination means that many parents can contribute to a single offspring, and uniform crossover means that each decision variable in an offspring is generated independently from every other decision variable. Combining global recombination and uniform crossover results in global uniform recombination. In addition, we suppose here that the entire population is used as potential contributors to each offspring, and fitness-based selection is used for each decision variable in each offspring. This results in GA with global uniform recombination (GA/GUR) as shown in Algorithm III.

Algorithm III. One generation of GA/GUR, where N is the population size. y and z are the entire population of candidate solutions, y_k is the kth candidate solution, and $y_k(s)$ is the sth decision variable of y_k. The crossover method is option (D) in Eq. (1b).

```
For each solution y_k, define parent probability μ_k proportional
  to fitness of y_k, μ_k∈[0,1]
z←y
For each solution z_k (k=1 to N)
    For each candidate solution decision variable index s
        Use {μ} to probabilistically select the parent solution y_j
        z_k(s)←y_j(s)
    Next candidate solution decision variable index
    Probabilistically decide whether to mutate z_k
Next solution
y←z
```

Comparing Algorithm II with Algorithm III, it is clearly seen that BBO reduces to GA/GUR when we set $\lambda_k=1$ for all k instead of setting λ_k as a variable in the BBO algorithm of Algorithm II. Also, GA/GUR can be viewed as a variation of BBO under special conditions. Since GA/GUR can be viewed as BBO variation, it follows that these two algorithms function identically under these special conditions.

2.1.3. Differential evolution

DE was introduced in 1997 (Storn and Price, 1997). It resembles the structure of evolution strategies, but differs from traditional evolution strategies in its generation of new candidate solutions and in its use of a greedy selection scheme. DE works as follows: first, all candidate solutions are randomly initialized and evaluated using the fitness function. Then for each candidate solution in the population, an offspring is created by adding the weighted difference of two parent solutions to the third. This process comprises one generation, and is executed as long as the termination condition is not fulfilled. The most basic form of DE is DE/rand/1/bin scheme, shown as

$$y_k(s) \leftarrow y_{r1}(s) + F(y_{r2}(s) - y_{r3}(s)) \tag{3}$$

where $r1$, $r2$, and $r3$ are three mutually exclusive random candidate solution indices, and F is the mutation scaling factor. A description of one generation of the basic DE algorithm is given in Algorithm IV.

Based on Algorithm IV, DE is conceptually similar to a combination of two ideas from GAs: global intermediate recombination and uniform crossover. As the name suggests in option (E) of (1b), global intermediate recombination means that a single offspring is produced by the weighted average of the two parent solutions, and uniform crossover as described above for GA/GUR. If we combine global intermediate recombination and uniform crossover, we obtain global uniform intermediate recombination. In addition, suppose that the entire population is used as potential contributors to each offspring, and fitness-based selection is used for the parent solutions. This results in GA with global uniform intermediate recombination (GA/GUIR) as shown in Algorithm V.

Algorithm IV. One generation of the basic DE algorithm, where N is the population size. y and z are the entire population of candidate solutions, y_k is the kth candidate solution, and $y_k(s)$ is the sth decision variable of y_k. CR is the probability of crossover, and F is the scaling factor.

```
z←y
For each solution z_k (k=1 to N)
    For each candidate solution decision variable index s
        Pick three random solutions y_r1, y_r2 and y_r3 that are
  distinct from each other as well as from solution z_k
        Pick a random index n between 1 and the
  population size
        Use CR (probabilistic) or n (deterministic) to
  decide on recombination
        If recombination then
            z_k(s)←y_r1(s) + F(y_r2(s)−y_r3(s))
End if
    Next candidate solution decision variable index
    Probabilistically decide whether to mutate z_k
Next solution
y←z
```

Algorithm V. One generation of GA/GUIR, where N is the population size. y and z are the entire population of candidate solutions, y_k is the kth candidate solution, and $y_k(s)$ is the sth decision variable of y_k, F is a weighted coefficient. The crossover method is option (E) in Eq. (1b).

```
For each solution y_k, define selection probability μ_k proportional
  to fitness of y_k, μ_k∈[0,1]
z←y
For each solution z_k (k=1 to N)
    For each candidate solution decision variable index s
        Use {μ} to probabilistically select the parent solutions
  y_r1 and y_r2
        z_k(s)←y_r1(s) + F(y_r2(s)−y_r1(s))
    Next candidate solution decision variable index
    Probabilistically decide whether to mutate z_k
Next solution
y←z
```

Comparing Algorithm IV with Algorithm V, it can be seen that DE is a generalization of GA/GUIR. Stated differently, GA/GUIR is a special case of DE. DE involves the selection of three random candidate solutions from the population, denoted $r1$, $r2$ and $r3$, and the generation of a random integer n between 1 and the population size. However, if $r1$ is selected on the basis of fitness, $r3$ is replaced with $r1$ and is selected on the basis of fitness, and $n=1$, then DE in Algorithm IV is equivalent to GA/GUIR as shown in Algorithm V. Namely, GA/GUIR can be viewed as a special case of DE. It therefore follows that these two algorithms perform identically under these special conditions.

2.1.4. Evolution strategies

ESs are optimization techniques based on the ideas of adaptation and evolution. They use natural problem-dependent representations, and primarily depend on mutation and selection as search operators. In the process of evolution, mutation is performed by adding a normally distributed random value to each solution decision variable. The step size or mutation strength is often governed by self-adaptation. The selection in ESs is deterministic and only based on the fitness rankings, not on the actual fitness values. The most popular ES is the (μ, λ)–ES: the μ parent solutions produce λ offspring solutions using mutation. Each of the λ offspring solutions is then assigned a fitness value depending on its quality. The best μ offspring solutions become the next generation parent solutions. This means λ must be greater or equal to μ. A description of one generation of the (μ, λ)–ES algorithm is given in Algorithm VI. Note that the μ and λ that are used in (μ, λ)–ES notation are not related to the μ and λ that are used in BBO notation.

Algorithm VI. One generation of the (μ, λ)–ES algorithm, where μ is the population size of the parent solutions, λ is the population size of the offspring, y and z are the entire population of candidate solutions, z_k is the kth candidate solution, and $z_k(s)$ is the sth decision variable of z_k, $\delta_k$is an adaptive mutation parameter, and $N_k(0,1)$ is a normally distributed random value with zero mean and unity variance.

```
z←y
For each solution z_k (k=1 to λ)
    For each candidate solution decision variable index s
        Randomly select two parents y_a(s) and y_b(s)
        Use a recombination method to combine the two
  parent features to obtain z_k(s)
        Update the adaptive mutation parameter δ_k
        z_k(s)←z_k(s) + δ_k N_k(0, 1)
    Next candidate solution decision variable index
Next solution
Evaluate each solution z_k (k=1 to λ), and save the fittest
  solutions as y_k (k=1 to μ, λ≥μ)
```

The similarities and differences between GAs and ESs are discussed in detail in Frank and Thomas (1991). Here we conceptually analyze the equivalences of the (μ, λ)–ES and GA/GUR. In

Algorithm VI, when the population size of the parent solutions μ is equal to the population size of the offspring solutions λ, the fitness values are used to select parents y_a and y_b, the entire population is used as potential contributors to the next generation, and a constant (non-adaptive) mutation parameter is used, the (μ, λ)–ES algorithm is equivalent to the GA with global uniform recombination (GA/GUR) described in Algorithm III. Namely, GA/GUR can be viewed as a (μ, λ)–ES algorithm under special conditions. Since GA/GUR can be viewed as a (μ, λ)–ES variation, it follows that these two algorithms function identically under these special conditions.

2.1.5. Particle swarm optimization

PSO was introduced in 1995 (Kennedy and Eberhart, 1995). It is inspired by the swarming behavior of a flock of birds and a school of fish. PSO consists of a swarm of particles moving in an n-dimensional search space of possible problem solutions. Every particle has a position vector y_k encoding a candidate solution to the problem (similar to the chromosome in GAs) and a velocity vector v_k to update position. Moreover, each particle contains a small memory that stores its own best position P_{best} and a global best position G_{best} obtained through communication with its neighbor particles. We use the fully connected network topology to transfer information, which means that G_{best} is the best position in the entire population at the current generation. Intuitively, the information about good solutions spreads through the swarm, and thus the particles tend to move to good areas in the search space. At each iteration, the velocity of a particle is updated based on its previous velocity, its own previous best position, and the current global best position, and is determined by

$$v_k(s) \leftarrow w v_k(s) + U(0,\phi_1)(P_{best}(s) - y_k(s)) + U(0,\phi_2)(G_{best}(s) - y_k(s)) \quad (4)$$

where $v_k(s)$ is the velocity of the kth particle in the sth dimension, w is called the inertia weight and controls the contribution of the previous velocity to the calculation of the new velocity, $U(a, b)$ is a uniformly distributed random number between a and b, and the cognitive constant ϕ_1 and the social constant ϕ_2 for neighborhood interaction determine the significance of $P_{best}(s)$ and $G_{best}(s)$ respectively. Furthermore, the position of the particle is calculated as the sum of the previous position and the new velocity:

$$y_k(s) \leftarrow y_k(s) + v_k(s) \quad (5)$$

A description of one generation of the basic PSO algorithm is given in Algorithm VII.

A simplified PSO algorithm can be obtained if each particle's velocity at each generation is independent of its previous velocity, namely, the inertia weight of the velocity w is set to 0; the proportionality constant ϕ_1 which determines the significance of $P_{best}(s)$ is set to 0; the global best position $G_{best}(s)$ is probabilistically selected based on fitness; and (4) and (5) are combined. In this case, Algorithm VII becomes Algorithm VIII.

Algorithm VII. One generation of the basic PSO algorithm, where N is the population size. y and z are the entire population of candidate solutions, z_k is the kth candidate solution, $z_k(s)$ is the sth decision variable of z_k, v_k is the kth particle's velocity (intermediate variable), $v_k(s)$ is the sth dimension of v_k, w is the inertia weight, $U(0, \phi)$ is a uniformly distributed random number between 0 and ϕ, P_{best} is the particle's previous best solution, and G_{best} is the current global best solution.

$z \leftarrow y$
For each solution (particle's position) z_k ($k=1$ to N)
  For each candidate solution decision variable index s
    $v_k(s) \leftarrow w v_k(s) + U(0,\phi_1)(P_{best}(s) - z_k(s)) + U(0,\phi_2)(G_{best}(s) - z_k(s))$
    $z_k(s) \leftarrow z_k(s) + v_k(s)$
  Next candidate solution decision variable index
  Probabilistically decide whether to mutate z_k
Next solution
$y \leftarrow z$

Algorithm VIII. One generation of a simplified PSO algorithm, where N is the population size, y and z is the entire population of candidate solutions, z_k is the kth candidate solution, $z_k(s)$ is the sth dimension of z_k, $U(0, \phi_2)$ is a uniformly distributed random number between 0 and ϕ_2, and G_{best} is the global best solution.

For each solution y_k, define selection probability μ_k proportional to fitness of y_k, $\mu_k \in [0,1]$
$z \leftarrow y$
For each solution z_k ($k=1$ to N)
  For each candidate solution decision variable index s
    Use $\{\mu\}$ to probabilistically select the global best solution $G_{best}(s)$
    $z_k(s) \leftarrow z_k(s) + U(0,\phi_2)(G_{best}(s) - z_k(s))$
  Next candidate solution decision variable index
  Probabilistically decide whether to mutate z_k
Next solution
$y \leftarrow z$

From Algorithm VIII, it is seen that a single offspring is recombined by the weighted average of its parent solution and the global best solution, where the weighting coefficient is $U(0, \phi_2)$, and each solution decision variable in an offspring is generated independently from every other solution decision variable. In addition, fitness-based selection is also used for each solution decision variable in each offspring. Comparing Algorithm VIII with Algorithm V, it is found that this PSO variation, like DE, is equivalent to a GA with global uniform intermediate recombination (GA/GUIR). Namely, GA/GUIR can be conceptually viewed as PSO under special conditions. Since GA/GUIR can be viewed as a simplified PSO algorithm, it follows that these two algorithms function identically under these special conditions.

2.1.6. Summary of EA similarities

We have seen above that several popular evolutionary algorithms including BBO, DE, ES and PSO, are conceptually similar to GAs. Under special conditions, these algorithms are equivalent to GA with global uniform recombination (GA/GUR). Note that GA with global uniform intermediate recombination (GA/GUIR) is a special case of GA with global uniform recombination (GA/GUR). All these algorithms have certain features in common, and they all adopt some operators to share information between solutions. Since they use many similar operators, it is not difficult to understand why they have nearly the same optimization ability in some real-world applications. We also recognize that there are many other well-established evolutionary algorithms, including evolutionary programming (EP) (Yao et al., 1999), the estimation of distribution algorithm (EDA) (Pedro and Lozano, 2002) and ant colony optimization (ACO) (Dorigo et al., 2002; Dorigo and Gambardella, 1997), and the study of their equivalences is deferred for future research.

2.2. Differences between EAs

The identical functionality of different EAs that we discussed above occurs only under special conditions, and each EA still has its own particular features and parameters that give it a unique

flexibility that other EAs may not have. In this subsection we point out some differences between these evolutionary algorithms.

First, their differences result from their unique biological motivations. GAs are based on survival of the fittest and genetically-motivated recombination strategies, BBO is based on the migration behavior of species between islands, DE uses candidate solution vector differences, ES uses self-adaptive mutation rates, and PSO is based on the foraging behavior of birds. It is therefore useful to retain the distinction between these EAs because they are based on different natural phenomena.

Retaining the biological foundation of GAs stimulates the incorporation of features from biology in GAs, which makes the study of GAs richer and more flexible. Some of these features include gender, niching, crowding, resource competition, aging, co-evolution, and ontogeny. Retaining BBO as a separate algorithm stimulates the incorporation of behaviors from natural biogeography into the BBO algorithm, including the effect of geographical proximity on migration rates, migration momentum, habitat area, and nonlinear migration curves (Ma, 2010). Retaining DE as a separate algorithm stimulates the incorporation of simple mathematical formulae into the DE algorithm, including arithmetic operation and vector computing (Das and Suganthan, 2011). Retaining ES as a separate algorithm stimulates the incorporation of various mutation methods, including the stochastic distribution and the covariance matrix of this distribution (Hansen, 2006). Retaining PSO as a separate algorithm stimulates the incorporation of swarming behaviors from birds, insects, or fish into the algorithm (Kennedy, 1997).

The second point in this section is that EA differences arise because the historical developments of the algorithms were different. We note that GAs and ES reproduce children by crossover; namely, their parent solutions disappear and are replaced by children at the end of each generation. BBO does not involve reproduction or the generation of children, and its solutions are not discarded after each generation, but are rather modified directly by migration. Like BBO, DE and PSO solutions are maintained from one generation to the next, and each solution is able to learn from its neighbors and adapt itself as the algorithm progresses. But PSO solutions change by virtue of another variable (velocity) and DE solutions change based on differences between other solutions.

Unifying various EAs is instructive, but retaining their own characteristics and distinctions provides rewarding mathematical and theoretical studies, and valuable tools for practical problems. From the no free lunch theorem (Wolpert and Macready, 1997), we know that if an algorithm achieves superior results on some problems, it must pay with inferior results on other problems. So the availability of various EAs provides a wealth of alternative optimization algorithms. Their differences provide opportunities for practical applications to a variety of problems, and for resulting contributions to the EA literature.

Table 1 summarizes the differences and similarities among the five algorithms that we consider, along with the conditions under which each algorithm reduces to a GA. Note that the year in which each algorithm was invented is not always clear-cut. For example, computer simulations of biological evolution were first conducted in the early 1950s (Barricelli, 1954), and the term "genetic algorithm" was not used until 1975 (Holland, 1975), but we can consider David Fogel's book in the early 1960s as the earliest work that closely resembles today's GAs (Fogel et al., 1966).

The "Primary Method of Obtaining Offspring" row in Table 1 indicates whether new candidate solutions are primarily created by modifying parent individuals, or by recombining parent individuals. This characteristic is not always clear-cut. For example, the (1+1)–ES does not use recombination, but the $(\mu+1)$–ES, $(\mu+\lambda)$–ES, and (μ,λ)–ES all use recombination. As another example, DE is a modification algorithm if $r1=k$ in Algorithm IV; but it is usually formulated with $r1\neq k$, in which case it is a recombination algorithm. In addition, GA, DE, and ES, which are categorized as "recombination" methods in Table 1, can also include modification through mutation. However, mutation is usually a low-probability occurrence, and often is restricted to changes of small magnitudes. In contrast, recombination typically occurs for every individual at every generation, and with no restrictions on the magnitude of the difference between the parents and children. Therefore, we categorize GA, DE, and ES, primarily as "recombination" algorithms.

The "Search Domain of Original Formulation" row in Table 1 indicates whether the algorithm was initially developed for binary or continuous search domains. This characteristic is not always clear-cut. For example, BBO was initially applied to binary search spaces (Simon, 2008), but its operation is more naturally suited to continuous search spaces, and it has generally been applied that way. In addition, GAs are usually applied to continuous search spaces, even though they were restricted to binary search spaces for their first couple of decades.

The "GA equivalence" row in Table 1 specifies what type of GA the algorithm is equivalent to: either a GA with global uniform recombination (GA/GUR—see Algorithm III), or a GA with global uniform intermediate recombination (GA/GUIR—see Algorithm IV). Finally, the "equivalence conditions" row in the table specifies the conditions under which the given algorithm is equivalent to either GA/GUR or GA/GUIR.

An examination of Table 1 can provide insights into which type of algorithm should be applied to a given problem. We begin by noting, as mentioned earlier, that all EAs can include mutation. Therefore, all of the EAs that we study here (GA, DE, ES, BBO, and PSO) can include the modification of existing solutions. However, the differences between these EAs can be characterized by noting that GA, DE, and ES primarily involve recombination, while BBO and PSO involve the modification of existing solutions. This indicates that GA, DE, and ES are more exploitative algorithms

Table 1
Summary of the similarities and differences among the five EAs.

	GA	BBO	DE	ES	PSO
Year Introduced	1966 (Fogel et al., 1966)	2008 (Simon, 2008)	1997 (Storn and Price, 1997)	1968 (Rechenberg, 1968)	1995 (Kennedy and Eberhart, 1995)
Primary method of obtaining offspring	Recombination	Modification	Recombination	Recombination	Modification
Search domain of original formulation	Binary	Continuous	Continuous	Continuous	Continuous
GA equivalence	–	GA/GUR	GA/GUIR	GA/GUR	GA/GUIR
Equivalence conditions	–	$\lambda=1$	$y_{r1}\sim$fitness $CR=1$	$\delta_k=0$ $y_a(s)=0$ $y_b(s)\sim$fitness	$w=0$ $\phi_1=0$ $G_{best}\sim$fitness

that may perform better in cases where the user can seed the initial population with known good candidate solutions, while BBO and PSO are more explorative algorithms that may perform better in cases where the user has less problem-specific information.

Second, note that although DE, ES, and PSO have all been applied to discrete search spaces, they were originally designed for continuous search spaces, so their application to combinatorial problems sometimes often seems unnatural. If a GA or BBO population is constrained to a discrete search space, then the next generation will also be so constrained. However, the same is not true for DE, ES, and PSO; for these algorithms, the offspring of a population that is constrained to a discrete search space will not itself necessarily be discrete. This observation is not intended as a criticism of DE, ES, and PSO for discrete search spaces, but it does indicate that procedural modifications and extra computational bookkeeping may be required to apply these algorithms to such domains. Although BBO is listed as "continuous" in Table 1, the "equivalence conditions" row in Table 1 indicates that it is more like a GA than the other algorithms. This indicates that BBO may perform better than DE, ES, or PSO on combinatorial problems.

3. Experimental results

This section first looks at the performance of the basic EAs, including GA, BBO, DE, ES and PSO, on a set of real-world optimization problems (Section 3.1). Then we compare the performance of advanced versions of these algorithms on the same set of real-world optimization problems (Section 3.2).

3.1. Comparison of the basic EAs

This subsection compares the performance of the basic GA, BBO, DE, ES and PSO described in the previous section on a set of real-world optimization problems from the 2011 IEEE Congress on Evolutionary Computation (Das and Suganthan, 2010). These functions are briefly summarized in Table 2. For the basic GA we use real coding, roulette wheel selection, single point crossover with a crossover probability of 1, which is a special case of option (B) in (1a), and a mutation probability of 0.001. For the basic BBO algorithm, we use a maximum immigration rate and maximum emigration rate of 1, linear migration curves as suggested in Simon (2008), and a mutation probability of 0. (For BBO mutation is beneficial primarily for small population sizes.) For the basic DE algorithm, we use a scaling factor F of 0.5, and a crossover rate CR of 0.5. For the (μ, λ)–ES algorithm, we use the population size of the parent solutions μ and the population size of the offspring solutions λ both equal to 50 each generation, and mutation parameter $\delta = 1$. For the basic PSO algorithm, we use an inertia weight w of 0.8, a cognitive constant ϕ_1 of 1.0, and a social constant ϕ_2 for neighborhood interaction of 1.0.

Each algorithm has a population size of 50, and a maximum of 100,000 fitness function evaluations. The granularity (that is, search space resolution) of each real-world optimization problem is 0.1, except for P01, P03, P10, P12 and P13, which are implemented with a granularity of 0.01. The results of solving these real-world optimization problems are given in Table 3. All results are computed from 25 independent simulations.

According to Table 3, the basic BBO algorithm performs best on 9 problems (P01, P02, P04, P06, P10, P11.2, P11.8, P11.9, and P11.10), the basic DE algorithm performs best on 6 problems (P07, P11.1, P11.4, P11.7, P12, and P13), the basic PSO algorithm performs best on 3 problems (P09, P11.5, and P11.6), and the basic GA algorithm performs best on problem P05. In addition, we see that for problems P03 and P08, all five algorithms attain the same optimum, and for problem P11.3, both the basic DE algorithm and the BBO algorithm attain the same optimum. These results indicate that the basic BBO algorithm is the most effective, the basic DE algorithm is the second most effective, and the basic PSO algorithm is the third most effective for the real-world benchmarks that we studied. This is because the basic BBO algorithm intelligently uses the fitness information of the solutions to determine the control parameters of each solution (the immigration rate and emigration rate).

In addition, the average running times of the five basic EAs are shown in the last row of Table 3. The basic GA algorithm is the fastest, and the basic BBO algorithm is the second fastest.

Table 4 shows the results of Friedman test comparisons for the basic GA, BBO, DE, ES and PSO. Friedman test is a nonparametric statistical method of the parametric two-way analysis of variance, which is a multiple comparisons test that aims to detect significant

Table 2
Problem set descriptions. More details about these problems can be found in Das and Suganthan (2010).

Problem	Dimension	Comments
P01	6	Parameter estimation for frequency-modulated (FM) sound waves.
P02	30	Lennard–Jones potential problem.
P03	1	Bifunctional catalyst blend optimal control problem.
P04	1	Optimal control of a nonlinear stirred tank reactor.
P05	30	Tersoff potential function minimization problem (instance 1).
P06	30	Tersoff potential function minimization problem (instance 2).
P07	20	Spread spectrum radar polyphase code design.
P08	7	Transmission network expansion planning problem.
P09	126	Large scale transmission pricing problem.
P10	12	Circular antenna array design problem.
P11.1	120	Dynamic economic dispatch problem (instance 1).
P11.2	216	Dynamic economic dispatch problem (instance 2).
P11.3	6	Static economic load dispatch problem (instance 1).
P11.4	13	Static economic load dispatch problem (instance 2).
P11.5	15	Static economic load dispatch problem (instance 3).
P11.6	40	Static economic load dispatch problem (instance 4).
P11.7	140	Static economic load dispatch problem (instance 5).
P11.8	96	Hydrothermal scheduling problem (instance 1).
P11.9	96	Hydrothermal scheduling problem (instance 2).
P11.10	96	Hydrothermal scheduling problem (instance 3).
P12	26	Spacecraft trajectory optimization problem (Messenger).
P13	22	Spacecraft trajectory optimization problem (Cassini2).

Table 3
Comparison of real-world optimization results for the basic GA, BBO, DE, ES and PSO. Here $[a \pm b]$ indicates the mean value and corresponding standard deviation of 25 independent simulations. The best result in each row is shown in bold font. In addition, CPU times (min) are shown in the last row of the table.

Prob.	GA	BBO	DE	ES	PSO
P01	9.19E−05 ± 4.24E−06	**7.35E−17 ± 2.53E−19**	4.51E−08 ± 2.47E−10	6.17E−02 ± 2.53E−03	7.68E−13 ± 1.95E−14
P02	−1.64E+01 ± 2.35E+00	**−2.83E+01 ± 1.27E+00**	−1.84E+01 ± 2.55E+00	−1.84E+01 ± 3.42E+00	−9.26E+00 ± 1.22E+00
P03	**1.15E−05 ± 0.00E+00**	**1.15E−05 ± 0.00E+00**	**1.15E−05 ± 0.00E+00**	**1.15E−05 ± 0.00E+00**	**1.15E−05 ± 0.00E+00**
P04	3.04E+01 ± 2.01E−01	**1.43E+01 ± 3.98E−01**	7.61E+01 ± 9.07E−01	9.45E+01 ± 3.74E−01	3.11E+01 ± 1.68E−01
P05	**−1.37E+01 ± 2.10E+00**	−9.20E+00 ± 5.65E+00	−6.63E+00 ± 1.84E+00	−3.69E+00 ± 1.87E+00	−1.14E+01 ± 2.94E+00
P06	−1.62E+01 ± 6.43E−01	**−2.92E+01 ± 1.86E−01**	−2.58E+01 ± 3.34E−01	−1.12E+01 ± 3.75E−01	−6.73E+00 ± 1.32E−01
P07	9.47E−01 ± 7.30E−03	9.71E−01 ± 3.64E−03	**8.27E−01 ± 1.60E−03**	9.07E−01 ± 7.48E−03	9.97E−01 ± 7.45E−03
P08	**2.20E+02 ± 0.00E+00**	**2.20E+02 ± 0.00E+00**	**2.20E+02 ± 0.00E+00**	**2.20E+02 ± 0.00E+00**	**2.20E+02 ± 0.00E+00**
P09	1.59E+03 ± 5.78E+00	1.04E+03 ± 2.55E+02	4.58E+03 ± 3.17E+01	1.89E+03 ± 6.91E+00	**3.51E+01 ± 1.87E+00**
P10	−1.18E+01 ± 6.63E+00	**−2.18E+01 ± 1.32E+00**	−1.84E+01 ± 6.62E+00	−4.25E+00 ± 1.17E+00	−1.89E+01 ± 3.92E+00
P11.1	5.78E+05 ± 1.64E+03	9.25E+04 ± 3.74E+03	**5.74E+04 ± 3.64E+03**	8.56E+05 ± 4.72E+03	2.36E+05 ± 4.42E+03
P11.2	6.24E+06 ± 5.57E+05	**1.05E+06 ± 1.96E+05**	2.79E+06 ± 1.32E+05	8.65E+06 ± 3.19E+05	2.68E+06 ± 3.14E+05
P11.3	1.56E+04 ± 7.19E+02	**1.54E+04 ± 2.89E+02**	**1.54E+04 ± 2.89E+02**	1.57E+04 ± 2.56E+02	1.56E+04 ± 2.89E+02
P11.4	7.26E+04 ± 3.20E+03	8.89E+04 ± 3.11E+03	**5.72E+04 ± 4.16E+03**	9.13E+04 ± 2.50E+03	6.87E+04 ± 4.95E+03
P11.5	7.05E+04 ± 2.24E+02	6.29E+04 ± 7.15E+02	7.91E+04 ± 3.58E+02	4.09E+04 ± 8.41E+02	**3.92E+04 ± 1.77E+02**
P11.6	3.47E+05 ± 6.38E+03	3.32E+05 ± 4.72E+03	4.16E+05 ± 8.47E+03	3.59E+05 ± 2.76E+03	**1.28E+05 ± 7.06E+03**
P11.7	4.85E+06 ± 7.64E+04	1.91E+06 ± 9.28E+04	**1.74E+06 ± 8.02E+05**	7.45E+06 ± 3.40E+04	3.24E+06 ± 5.79E+04
P11.8	1.88E+06 ± 6.35E+04	**9.23E+05 ± 1.02E+04**	1.23E+06 ± 6.68E+04	3.59E+06 ± 5.81E+04	6.37E+06 ± 4.41E+05
P11.9	3.50E+06 ± 4013E+04	**9.30E+05 ± 1.73E+04**	1.58E+06 ± 7.65E+04	7.14E+06 ± 7.63E+04	9.82E+05 ± 1.55E+04
P11.10	2.38E+06 ± 3.36E+04	**9.24E+05 ± 1.70E+03**	1.75E+06 ± 5.50E+04	1.12E+06 ± 1.81E+04	8.49E+06 ± 2.54E+04
P12	1.58E+01 ± 4.85E−01	1.64E+01 ± 4.81E+00	**9.34E+00 ± 1.78E−01**	6.88E+01 ± 5.32E−01	7.23E+01 ± 1.14E+00
P13	8.77E+00 ± 6.53E−02	1.43E+01 ± 1.78E+00	**8.42E+00 ± 9.46E−01**	9.57E+00 ± 5.81E−02	4.12E+01 ± 3.62E+00
Time	74.92	83.75	110.39	93.02	97.65

Table 4
Friedman test results for the basic GA, BBO, DE, ES and PSO. Here "Average" indicates Friedman average rank, and "Statistic" and "*p*-values" indicates Friedman statistic and corresponding *p*-values respectively.

Prob.	GA	BBO	DE	ES	PSO
P01	4	1	3	5	2
P02	4	1	2.5	2.5	5
P03	3	3	3	3	3
P04	2	1	4	5	3
P05	1	3	4	5	2
P06	3	1	2	4	5
P07	3	2	1	5	4
P08	3	3	3	3	3
P09	3	2	5	4	1
P10	4	1	3	5	2
P11.1	4	2	1	5	3
P11.2	4	1	3	5	2
P11.3	3.5	1.5	1.5	5	3.5
P11.4	3	4	1	5	2
P11.5	3	4	5	2	1
P11.6	3	2	5	4	1
P11.7	4	2	1	5	3
P11.8	3	1	2	4	5
P11.9	4	1	3	5	2
P11.10	4	1	3	2	5
P12	2	3	1	4	5
P13	5	3	1	2	4
Average	3.30	1.98	2.64	4.07	3.02
Statistic	21.690		*p*-value	0.00062	

differences between the behaviors of two or more algorithms (Derrac et al., 2011). We find that for the real-world benchmarks that we studied, the basic BBO algorithm is the best with an average rank of 1.98, the basic DE algorithm is the second best with an average rank of 2.64, and the basic PSO algorithm is the third best with an average rank of 3.02. Such results are consistent to those shown in Table 3. We also obtain Friedman statistic of 21.690 and corresponding *p*-value of 0.00062 based on the Friedman rank, where the detail of calculating procedure refers to literature (Derrac et al., 2011). Because the *p*-value is smaller than 0.05 (which is often used as the significance level or critical *p*-value), the result strongly indicates the existence of significant differences among the algorithms considered.

There are several reasons that we do not want to use our benchmark results to draw conclusions that are too definite. First, for EAs, different tuning parameter values might result in significant changes in their performance. This is mainly due to the difficulty in determining the optimum tuning parameters, such as population size and mutation rate. A small change in a tuning parameter could change the effectiveness of the algorithm. Second, if we use more advanced versions of GA, BBO, DE, ES and PSO, it might be possible to obtain better results than those here (as we see in the following section). The purpose of our comparisons is not to tune the control parameters of the algorithms to obtain the best possible performance, but rather to show that the differences of EAs result in different optimization performance. Third, we have discretized each optimization problem in this section and executed discrete versions of the EAs. We might obtain different results if we used continuous optimization problems.

3.2. Comparison of advanced EAs

The next experiment compares the performance of advanced versions of EAs, which generally provide better performance than the basic algorithms. These advanced EAs include the stud GA (SGA) with single-point crossover (a special case of option (B) in (1a)) (Khatib and Fleming, 1998), oppositional BBO (OBBO) (Ergezer and Simon, 2011; Ergezer et al., 2009), self-adaptive DE (SaDE) (Qin et al., 2009; Zhang and Sanderson, 2009; Zhao et al., 2011), covariance matrix adaptation ES (CMA-ES) (Beyer and Sendhoff, 2008; Hansen et al., 2003; Hansen et al., 2011), and standard PSO 2007 (SPSO 07) (Bratton and Kennedy, 2007; Particle Swarm Central). We select SGA because it is an improvement of the basic GA and uses the best individual at each generation for crossover. We select OBBO because it is one of the leading BBO variants and has obtained better performance than BBO on benchmark functions and real-world optimization problems (Ergezer et al., 2009). We select SaDE because it is one of the most powerful evolutionary algorithms and has demonstrated excellent performance on many problems. We select CMA-ES because it is the most successful improved variant of ES. Finally, we select SPSO 07

because it often offers good performance and is a relatively new PSO variation.

For SGA and OBBO, the parameters used in this subsection are the same as those in the previous subsection. For SaDE, the control parameter settings are gradually adapted according to the learning progress. The scaling factor F is randomly sampled from the normal distribution $N(0.5, 0.3)$ and the crossover rate CR follows the normal distribution $N(0.5, 0.1)$. For CMA-ES, the parameter settings can be shown in Hansen (2006) in detail.

For SPSO 07 we use an inertia weight of 0.8, a cognitive constant of 0.5, a social constant for swarm interaction of 1.0, and a social constant for neighborhood interaction of 1.0. Each algorithm has a population size of 50, and a maximum of 100,000 fitness function evaluations. The results are given in Table 5. All results are computed from 25 independent simulations.

According to Table 5, SaDE performs best on 6 problems (P07, P11.1, P11.4, P11.7, P12 and P13), CMA-ES performs best on 5 problems (P02, P10, P11.2, P11.9, and P11.10), OBBO performs best on 2 problems (P01, P11.6), SPSO 07 performs best on problem P04, and SGA performs best on problem P11.5. In addition, we see that for problems P03, P08, P11.3, P11.8, all five algorithms attain the same optimum; for problem P05, SaDE, CMA-ES and SGA attain the same optimum; for problem P06, OBBO, SaDE and CMA-ES attain the same optimum; and for problem P09, both SaDE and SPSO 07 attain the same optimum. These results show that SaDE performs similarly to CMA-ES, and is significantly better than OBBO, SPSO 07 and SGA for the real-world optimization problems.

These conclusions are completely different than those of the basic EAs in the previous section. This is because the more advanced versions of GA, BBO, DE, ES and PSO adaptively tune the control parameters to obtain better results. For example, control parameter settings in SaDE are gradually adapted according to the learning progress. The covariance matrix adaptation in CMA-ES adapts the covariance matrix of a multivariate normal search distribution. The results further indicate that although these EAs are conceptually equivalent under special conditions, they have different optimization performances because of differences in implementation details. Note that different comparisons of advanced versions of GA, BBO, DE, ES and PSO might result in different performance rankings than we found. There are many advanced forms of these algorithms, and researchers are continually proposing new variants in their search for improved algorithms.

The average running times of the five advanced EAs are shown in the last row of Table 5. SGA is the fastest algorithm, and OBBO is the second fastest.

Table 6 shows the results of Friedman test comparisons for SGA, OBBO, SaDE, CMA-ES and SPSO 07. We find that SaDE is the best with an average rank of 2.38, CMA-ES is the second best with an average rank of 2.41, and OBBO is the third best with an average

Table 6
Friedman test results for SGA, OBBO, SaDE, CMA-ES and SPSO 07. Here "Average" indicates Friedman average rank, and "Statistic" and "p-values" indicates Friedman statistic and corresponding p-values respectively.

Prob.	SGA	OBBO	SaDE	CMA-ES	SPSO 07
P01	4	1	2	3	5
P02	4	3	2	1	5
P03	3	3	3	3	3
P04	4	3	2	5	1
P05	2	5	2	2	4
P06	4	2	2	2	5
P07	4	5	1	2	3
P08	3	3	3	3	3
P09	4	5	1.5	3	1.5
P10	5	2	3	1	4
P11.1	2	5	1	3	4
P11.2	5	2	4	1	3
P11.3	3	3	3	3	3
P11.4	3	2	1	4	5
P11.5	1	4	5	2	3
P11.6	5	1	4	2	3
P11.7	5	2	1	3	4
P11.8	3	3	3	3	3
P11.9	5	2	4	1	3
P11.10	4	2	3	1	5
P12	2	5	1	3	4
P13	4	5	1	2	3
Average	3.60	3.09	2.38	2.41	3.52
Statistic	12.064		p-values	0.03148	

Table 5
Comparison of real-world optimization results for SGA, OBBO, SaDE, CMA-ES and SPSO 07. Here $[a \pm b]$ indicates the mean value and corresponding standard deviation. The best result in each row is shown in bold font. In addition, CPU times (min) are shown in the last row of the table.

Prob.	SGA	OBBO	SaDE	CMA-ES	SPSO 07
P01	7.44E−18 ± 5.11E−19	**0.00E+00 ± 0.00E+00**	1.34E−19 ± 3.17E−20	2.94E−18 ± 3.18E−19	2.60E−02 ± 4.83E−03
P02	−2.62E+01 ± 1.75E+00	−2.81E+01 ± 1.39E+00	−2.88E+01 ± 5.66E+00	**−2.90E+01 ± 5.32E+00**	−1.25E+01 ± 1.53E+00
P03	**1.15E−05 ± 0.00E+00**	**1.15E−05 ± 0.00E+00**	**1.15E−05 ± 0.00E+00**	**1.15E−05 ± 0.00E+00**	**1.15E−05 ± 0.00E+00**
P04	2.87E+01 ± 3.26E−01	2.12E+01 ± 8.45E−01	2.03E+01 ± 6.72E−01	3.58E+01 ± 3.10E−01	**1.37E+01 ± 4.85E−01**
P05	**−3.70E+01 ± 9.06E+00**	−2.20E+01 ± 3.44E+00	**−3.70E+01 ± 9.06E+00**	**−3.70E+01 ± 9.06E+00**	−2.27E+01 ± 2.01E+00
P06	−2.91E+01 ± 8.64E−01	**−2.94E+01 ± 5.48E−01**	**−2.94E+01 ± 5.48E−01**	**−2.94E+01 ± 5.48E−01**	−2.68E+01 ± 6.77E−01
P07	6.89E−01 ± 6.25E−03	7.14E−01 ± 2.25E−03	**4.93E−01 ± 4.87E−03**	5.00E−01 ± 1.22E−02	5.08E−01 ± 7.65E−03
P08	**2.20E+02 ± 0.00E+00**	**2.20E+02 ± 0.00E+00**	**2.20E+02 ± 0.00E+00**	**2.20E+02 ± 0.00E+00**	**2.20E+02 ± 0.00E+00**
P09	3.05E+02 ± 4.64E+01	1.63E+03 ± 9.31E+01	**4.60E+01 ± 5.64E+00**	9.47E+01 ± 5.14E+00	**4.60E+01 ± 5.64E+00**
P10	−2.01E+01 ± 7.65E+00	−2.35E+01 ± 1.54E+00	−2.28E+01 ± 9.36E+00	**−3.24E+01 ± 4.36E+00**	−2.09E+01 ± 4.18E+00
P11.1	4.79E+04 ± 5.87E+02	6.10E+05 ± 3.26E+04	**1.14E+04 ± 2.56E+03**	8.14E+04 ± 9.35E+02	1.50E+05 ± 1.49E+04
P11.2	1.81E+07 ± 7.26E+05	3.42E+06 ± 2.94E+05	7.34E+06 ± 4.25E+05	**9.48E+05 ± 6.24E+04**	6.10E+06 ± 4.27E+05
P11.3	**1.54E+04 ± 2.86E+02**	**1.54E+04 ± 2.86E+02**	**1.54E+04 ± 2.86E+02**	**1.54E+04 ± 2.86E+02**	**1.54E+04 ± 2.86E+02**
P11.4	1.80E+04 ± 4.55E+03	1.57E+04 ± 4.58E+03	**9.21E+03 ± 7.36E+02**	1.82E+04 ± 2.40E+03	2.47E+04 ± 8.45E+03
P11.5	**3.20E+04 ± 7.11E+02**	5.85E+04 ± 1.23E+02	9.52E+04 ± 4.31E+02	3.26E+04 ± 4.37E+02	5.48E+04 ± 3.22E+02
P11.6	6.12E+05 ± 7.49E+03	**1.15E+05 ± 8.92E+03**	2.08E+05 ± 5.21E+03	1.21E+05 ± 3.92E+03	1.37E+05 ± 3.55E+03
P11.7	2.27E+06 ± 3.14E+04	1.81E+06 ± 7.06E+05	**1.14E+06 ± 1.39E+04**	1.95E+06 ± 3.19E+04	2.13E+06 ± 1.03E+05
P11.8	**9.20E+05 ± 0.00E+00**	**9.20E+05 ± 0.00E+00**	**9.20E+05 ± 0.00E+00**	**9.20E+05 ± 0.00E+00**	**9.20E+05 ± 0.00E+00**
P11.9	9.39E+05 ± 4.33E+04	1.10E+06 ± 2.07E+04	5.22E+06 ± 1.27E+04	**8.58E+05 ± 4.66E+04**	1.60E+06 ± 2.01E+04
P11.10	1.63E+06 ± 6.13E+04	9.51E+05 ± 6.83E+03	1.21E+06 ± 2.95E+04	**9.01E+05 ± 480E+03**	4.58E+06 ± 6.21E+04
P12	7.89E+00 ± 3.12E−01	1.57E+01 ± 8.96E−01	**6.74E+00 ± 3.17E−01**	9.14E+00 ± 5.47E−01	1.38E+01 ± 6.90E−01
P13	3.21E+01 ± 6.79E−02	3.97E+01 ± 5.24E−01	**6.63E+00 ± 8.92E−01**	8.65E+00 ± 7.61E−01	8.78E+00 ± 1.54E−01
Time	73.28	87.34	124.92	135.65	101.20

rank of 3.09. Such results are consistent with those shown in Table 5. We also obtain a Friedman statistic of 12.064 and corresponding p-value of 0.03148 based on the Friedman rank, which is smaller than 0.05. The result indicates the existence of significant differences among the algorithms considered.

4. Conclusions

In this paper the equivalences and differences of various popular population-based EAs, including GA, BBO, DE, ES and PSO, are discussed in detail based on algorithm motivations and implementation details. Because these algorithms have so many similarities due to their reliance on organic evolution, it is found that the basic versions of BBO, DE, ES and PSO are equivalent to the GA with global uniform recombination (GA/GUR) under certain conditions. In addition, we compared the basic versions and the advanced versions of GA, BBO, DE, ES and PSO on a set of real-world continuous optimization benchmark problems, and empirical results show that although these algorithms are nearly identical under special conditions, their optimization performances are different with the conditions that we tested, because they retain their own characteristics when implemented in their standard forms. Although many EAs are equivalent under special conditions, we conclude that it is necessary to maintain the distinction between various EAs, because they provide a wealth of optimization algorithm development and application opportunities.

EA researchers and practitioners often want to know which algorithm is best. However, the no free lunch theorem (Wolpert and Macready, 1997) and the empirical results in this paper show that this question is moot. These results show that relative performance is greatly affected by the variant or tuning parameters of the EAs. These results also support the general theoretical contribution of the paper: the EAs are equivalent under certain conditions.

For future work there are several important directions. This paper generalizes the equivalences and differences of several mainstream EAs. But many other algorithms exist, including ACO, EDA, ABC, AIS, and their variants, which may provide better optimization performance for certain classes of problems than the algorithms in this paper. So it is of interest to discuss and analyze their equivalences and differences also. The second important direction for future work is to study theoretical equivalences and differences of these algorithms based on Markov chains, dynamic systems, statistical mechanics, or other mathematical models. This will provide more definite theoretical conclusions. The third important direction for future work is to develop better state-of-the-art versions of these EAs by using natural principles. Finally, we note that our empirical comparisons have been restricted to optimization problems with continuous domains. It will be interesting and important in future research to compare the performance of various EAs on a standard set of combinatorial benchmarks (for example, traveling salesman problems).

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant nos. 61074032, 61179041, the Zhejiang Provincial Natural Science Foundation of China under Grant no. Y1090866, and the Project of Science and Technology Commission of Shanghai Municipality under Grant no. 10JC1405000. The comments of the reviewers were instrumental in strengthening this paper from its first version.

References

Ahn, C., 2006. Advances in Evolutionary Algorithms: Theory, Design and Practice. Springer Publishing, New York.
Arnold, D.V., Beyer, H.G., 2001. Investigation of the (μ, λ)-ES in the presence of noise. In: Proceedings of IEEE Congress on Evolutionary Computation, Seoul, Korea, pp. 332–339.
Barricelli, N., 1954. Esempi numerici di processi di evoluzione. Methodos 6, 45–68. (The English translation of the title is "Numerical models of evolutionary processes.").
Beyer, H.G., 1994. Toward a theory of evolution strategies: The (μ, λ)-theory. Evol. Comput. 2 (4), 381–407.
Beyer, H.G., Sendhoff, B., 2008. Covariance matrix adaptation revisited—the CMSA evolution strategy. Lect. Notes Comput. Sci. 5199, 123–132.
Bratton, D., Kennedy, J., 2007. Defining a standard for particle swarm optimization. In: Proceedings of IEEE Swarm Intelligence Symposium, Honolulu, Hawaii, 2007, pp. 120–127.
Chiong, R., Weise, T., Michalewicz, Z., 2012. Variants of Evolutionary Algorithms for Real-World Applications. Springer, New York.
Das, S., Suganthan, P.N., 2010. Problem Definitions and Evaluation Criteria for CEC 2011 Competition on Testing Evolutionary Algorithms on Real World Optimization Problems. Technical Report. Jadavpur University, Nanyang Technological University.
Das, S., Suganthan, P.N., 2011. Differential evolution—a survey of the state-of-the-art. IEEE Trans. Evol. Comput. 15 (1), 4–31.
Derrac, J., Garcia, S., Molina, D., Herrera, F., 2011. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol. Comput. 1 (1), 3–18.
Dorigo, M., Gambardella, L.M., 1997. Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans. Evol. Comput. 1 (1), 53–66.
Dorigo, M., Maniezzo, V., Colorni, A., 2002. Ant system: Optimization by a colony of cooperating agents. IEEE Trans. Syst. Man Cybern.—Part B 26 (1), 29–41.
Eberhart, R., Shi, Y., 1998. Comparison between genetic algorithms and particle swarm optimization. In: International Conference on Evolutionary Programming, San Diego, CA, March 1998, pp. 611–616.
Elbeltagi, E., Hegazy, T., Grierson, D., 2005. Comparison among five evolutionary-based optimization algorithms. Adv. Eng. Inform. 19 (1), 43–53, Jan..
Ergezer, M., Simon, D., 2011. Oppositional biogeography-based optimization for combinatorial problems. In: IEEE Congress on Evolutionary Computation, New Orleans, LA, pp. 1496–1503, 2011.
Ergezer, M., Simon, D., Du, D., 2009. Oppositional biogeography-based optimization. In: Proceedings of the IEEE Conference on Systems, Man, and Cybernetics, San Antonio, TX, 2009, pp. 1035–1040.
Fogel, L., Owens, A., Walsh, M., 1966. Artificial Intelligence through Simulated Evolution. John Wiley & Sons, Hoboken, NJ.
Frank, H., Thomas, B., 1991. Genetic algorithms and evolution strategies: Similarities and differences. Lect. Notes Comput. Sci. 496, 455–469.
Gao, W., 2004. Comparison study of genetic algorithm and evolutionary programming. In: International Conference on Machine Learning and Cybernetics, Shanghai, China, August 2004, pp. 204–209.
Hansen, N., 2006. The CMA evolution strategy: a comparing review. In: Lozano, J.A., Larrañaga, P., Inza, I., Bengoetxea, E. (Eds.), Towards a New Evolutionary Computation: Advances in Estimation of Distribution Algorithms. Springer, New York, pp. 75–102.
Hansen, N., Müller, S.D., Koumoutsakos, P., 2003. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). Evol. Comput. 11 (1), 1–18.
Hansen, N., Ros, R., Mauny, N., Schoenauer, M., Auger, A., 2011. Impacts of invariance in search: when CMA-ES and PSO face ill-conditioned and non-separable problems. Appl. Soft Comput. 11 (8), 5755–5769.
Hoffmeister, F., Bäck, T., 1990. Genetic algorithms and evolution strategies: similarities and differences. First Workshop on Parallel Problem Solving from Nature, Dortmund, Germany, October 1990, pp. 455–469.
Hofmeyr, S., Forrest, S., 2000. Architecture for an artificial immune system. Evol. Comput. 8 (4), 443–473.
Holland, J., 1975. Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor, MI.
Karaboga, D., Basturk, B., 2007. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. J. Glob. Optim. 39 (3), 459–471.
Kennedy, J., 1997. The particle swarm: Social adaptation of knowledge. In: Proceedings of IEEE Conference on Evolutionary Computation, Indianapolis, IN, 1997, pp. 303–308.
Kennedy, J., Eberhart, R., 1995. Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, Perth, WA, 1995, pp. 1942–1948.
Khatib, W., Fleming, P., 1998. The stud GA: a mini revolution?. In: Eiben, a., Back, T., Schoenauer, M., Schwefel, H. (Eds.), Parallel Problem Solving from Nature. Springer, New York, pp. 683–691.
Lai, L., Sichanie, A., Gwyn, B., 1998. Comparison between evolutionary programming and a genetic algorithm for fault-section estimation. In: IEE Proceedings—Generation, Transmission and Distribution, vol. 145 (5), September 1998, pp. 616–620.
Ma, H., 2010. An analysis of the equilibrium of migration models for biogeography-based optimization. Inform. Sci. 180 (18), 3444–3464.

Merz, P., Freisleben, B., 1999. A comparison of memetic algorithms, tabu search, and ant colonies for the quadratic assignment problem. Congress on Evolutionary Computation, Washington, DC, July 1999, pp. 2063–2070.
Oltean, M., 2007. Evolving evolutionary algorithms with patterns. Soft Comput. 11 (6), 503–518.
Particle Swarm Central. ⟨http://www.particleswarm.info/⟩.
Pedro, L., Lozano, J.A., 2002. Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Kluwer Academic Publishers, Norwell, MA.
Qin, A.K., Huang, V.L., Suganthan, P.N., 2009. Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans. Evol. Comput. 13 (2), 398–417.
Rechenberg, I., 1968. Cybernetic solution path of an experimental problem. In: Fogel, D. (Ed.), Evolutionary Computation: The Fossil Record. Wiley-IEEE Press, Hoboken, NJ, pp. 301–310.
Reeves, C., Rowe, J., 2003. Genetic Algorithms: Principles and Perspectives. kluwer Academic Publisher, Norwell, MA.
Schwefel, H.P., 1995. Evolution and Optimum Seeking. Wiley Press, Hoboken, NJ.
Settles, M., Rodebaugh, B., Soule, T., 2003. Comparison of genetic algorithm and particle swarm optimizer when evolving a recurrent neural network. In: Proceedings of the Genetic and Evolutionary Computation Conference, Chicago, IL, July 2003, pp. 148–149.
Simon, D., 2008. Biogeography-based optimization. IEEE Trans. Evol. Comput. 12 (6), 702–713.
Simon, D., Rarick, R., Ergezer, M., Du, D., 2011. Analytical and numerical comparisons of biogeography-based optimization and genetic algorithms. Inform. Sci. 181 (7), 1224–1248.
Storn, R., Price, K.V., 1997. Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J. Glob. Optim. 11 (4), 341–359.
Vose, M.D., 1999. The Simple Genetic Algorithm: Foundations and Theory. The MIT press, Cambridge, MA.
Whitley, D., 2001. An overview of evolutionary algorithms: Practical issues and common pitfalls. Inform. Software Technol. 43 (14), 817–831.
Wolpert, D.H., Macready, W.G., 1997. No free lunch theorems for optimization. IEEE Trans. Evol. Comput. 1 (1), 67–82.
Yao, X., Liu, Y., Lin, G., 1999. Evolutionary programming made faster. IEEE Trans. Evol. Comput. 3 (2), 82–102.
Zhang, J.Q., Sanderson, A.C., 2009. JADE: Adaptive differential evolution with optional external archive. IEEE Trans. Evol. Comput. 13 (5), 945–958.
Zhao, S.Z., Suganthan, P.N., Das, S., 2011. Self-adaptive differential evolution with multi-trajectory search for large-scale optimization. Soft Comput. 15 (11), 2175–2185.