Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 456232, 18 pages
http://dx.doi.org/10.1155/2013/456232

Research Article

Complex System Optimization Using Biogeography-Based Optimization

Dawei Du and Dan Simon

Cleveland State University, 2121 Euclid Avenue, Cleveland, OH 44115, USA

Correspondence should be addressed to Dan Simon; d.j.simon@csuohio.edu

Received 14 August 2013; Revised 2 October 2013; Accepted 3 October 2013

Academic Editor: Oleg V. Gendelman

Complex systems are frequently found in modern industry. But with their multisubsystems, multiobjectives, and multiconstraints, the optimization of complex systems is extremely hard. In this paper, a new algorithm adapted from biogeography-based optimization (BBO) is introduced for complex system optimization. BBO/Complex is the combination of BBO with a multiobjective ranking system, an innovative migration approach, and effective diversity control. Based on comparisons with three complex system optimization algorithms (multidisciplinary feasible (MDF), individual discipline feasible (IDF), and collaborative optimization (CO)) on four real-world benchmark problems, BBO/Complex demonstrates competitive performance. BBO/Complex provides the best performance in three of the benchmark problems and the second best in the fourth problem.

1. Introduction

With the recent advances of technology in industry, many systems include more components and parts than those in the past. Such systems are more complex than ever before. The design optimization of such systems becomes more difficult under these circumstances. One familiar example is the design of the modern aircraft, where thousands of components need to be designed, and millions of parts need to be chosen for assembly. Due to the huge number of variables, it is extremely difficult to find an effective optimization method.

In the remainder of this section, we give a brief introduction to complex systems, optimization algorithms for complex systems, and biogeography-based optimization (BBO). In Section 2, we introduce the new optimization algorithm for complex systems (BBO/Complex). Section 3 demonstrates the performance of BBO/Complex with competitor algorithms. Section 4 presents conclusions and plans for future work.

1.1. Complex Systems. According to [1], a complex system has the following properties: (1) a complex system contains a large number of elements; (2) the elements have interactions with each other; (3) the interactions are rich; (4) the interactions include certain characteristics such as nonlinearity. In [2], a complex system is defined as "[a]*n* assembly of interacting members that is difficult to understand as a whole." We see that complex systems can have various structures, as long as they satisfy the above descriptions. Considering these descriptions and real-world systems in modern industry, we propose here that a complex system includes the following characteristics: (1) multiple objectives; (2) multiple constraints; (3) multiple variables; (4) high degree of nonlinearity. This is an ambiguous and fuzzy definition, but no more so than the definitions of many other engineering terms. Perhaps it is appropriate that the definition of a complex system is, itself, complex.

The mathematical description of a system comprises equations and inequalities that include the definitions of variables, the ranges of variables, and the connections between variables. Optimizing a system is equivalent to mathematically defining the system and then finding the feasible solutions that (approximately) optimize the objective functions. But when the order of the equations or inequalities is relatively large or those equations or inequalities are highly

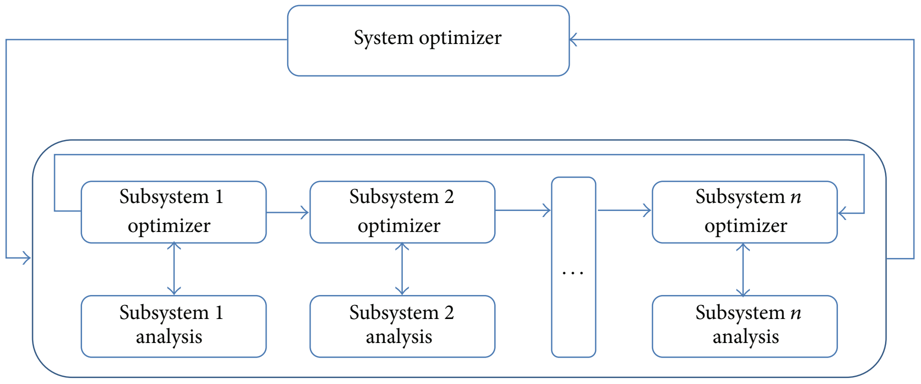

Figure 1: Multidisciplinary feasible (MDF) formulation [9].

nonlinear, the solutions must be obtained numerically rather than analytically [3]. Unfortunately, most complex systems include interacting subsystems that are either continuous or NP-hard and thus contain a huge number of possible solutions. The inclusion of subsystems in complex systems adds even more complexity than that involved in a single system.

1.2. The Optimization of Complex Systems: Multidisciplinary Design Optimization. Multidisciplinary design optimization (MDO) is a class of optimization methods dedicated to solving design problems that involve more than one discipline. Its definition is as follows: "Multidisciplinary Design Optimization (MDO) is a methodology for design and analysis of complex engineering systems and subsystems which coherently exploits the synergism of mutually interacting phenomena" [4]. Based on this definition, we see that MDO algorithms are good candidates for complex system optimization tools.

In the 1970s and 1980s, computer aided design became a mature approach for aircraft design, including economic factors, manufacturability, and reliability. Aircraft design was the initial motivation of MDO [5]. With thousands of parts and parameters in airplane design, MDO provided a revolution in the aircraft industry. In 1989, the American Institute of Aeronautics and Astronautics (AIAA) established the technical committee on MDO [5].

As mentioned above, MDO is a class of optimization methods. Numerous algorithms belong to this class, such as multidisciplinary feasible (MDF), which is the most popular MDO algorithm [6]; individual discipline feasible (IDF), which does not require system decomposition [7]; and collaborative optimization (CO), which is effective for many complex systems and which has been widely adopted in industry [8].

Traditional MDO algorithms are frameworks that provide basic conceptual structures without specifying the detailed underlying algorithms. In [9], the definition of MDO is given as follows: "an MDO method for a given problem consists of an MDO formulation and an optimization algorithm." The particular optimization algorithm is usually chosen based on the specific problem or the user's preference. Different MDO methods can share the same underlying optimization algorithm. Conversely, the same MDO method can be implemented with different underlying optimization algorithms. Therefore, the major difference between MDO algorithms is the MDO formulation, or, in other words, the structure of the method.

The most popular MDO algorithms include MDF, CO, and IDF. MDF is perhaps the most well-known MDO algorithm. It is often considered the standard solution method for multidisciplinary problems. The structure of a typical MDF algorithm is shown in Figure 1. The top level of MDF is system optimization. The second level is called multidisciplinary analysis (MDA), which passes coupled variables among subsystems to obtain feasible solutions at the subsystem level after a certain number of iterations. After reaching the iteration limit, the second level passes its solution to the first level, and this completes one optimization cycle. The iteration cycle limit is usually defined by the user. The structure of MDF enables it to be a very competitive optimization method when the subsystems are highly coupled.

CO is another typical MDO algorithm and has a bilevel structure which is shown in Figure 2. The first level is the system optimizer, which optimizes the feedback from the subsystem optimizers. The second level is the combination of the subsystem optimizers, which optimize each subsystem. Unlike MDF, the subsystem optimizations in CO are independent from each other, which means that CO puts more focus on subsystem optimization, which is advantageous for systems with extremely complex subsystems that are loosely coupled.

IDF is an all-in-one MDO algorithm. The most significant benefit of IDF is that it can optimize all of the subsystems together without subsystem optimizations. For most MDO algorithms, decomposition of the system is necessary. But unlike CO, IDF does not require subsystem optimization. It treats subsystems more like objective functions. As long as we have the objectives and constraints for each subsystem, IDF can be implemented. As we see from the structure of IDF in Figure 3, IDF includes subsystem analysis but not subsystem optimizers, which makes it an all-in-one

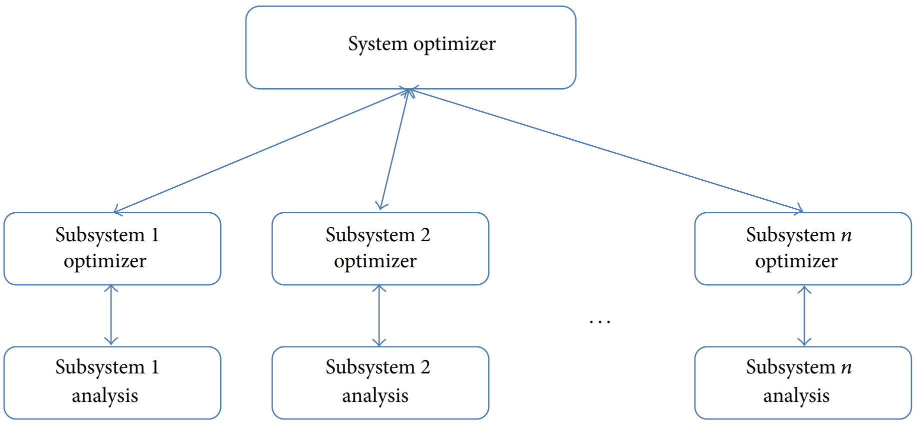

Figure 2: Collaborative optimization (CO) [9].

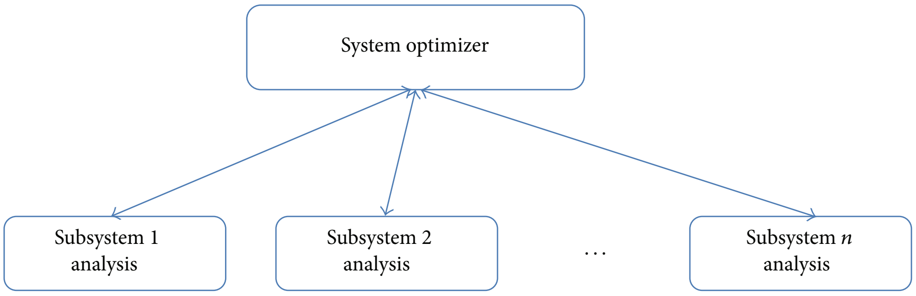

Figure 3: Individual discipline feasible (IDF) formulation [9].

algorithm. Optimization only operates at the global system level.

1.3. Biogeography-Based Optimization. BBO is a newly invented heuristic algorithm that was first introduced in 2008 [10]. Like most heuristic algorithms, it is inspired by nature. The inspiration of BBO comes from the distribution of species over time and area. The environment of BBO is analogous to an archipelago of islands, and each island is considered as a possible solution to the problem. Each decision variable is called a suitability index variable (SIV) in BBO, and each island consists of SIVs. The performance of each island is measured by objective functions, and we use habitat suitability index (HSI) to represent the level of performance. BBO uses migration to share SIVs and mutation to randomly create new SIVs. The basic procedure of the BBO algorithm is as follows [11].

(1) Define the mutation probability and the elitism parameter. Mutation and elitism are the same as in genetic algorithms and many other evolutionary algorithms [12].

(2) Initialize the population. Again, this is performed as in any other evolutionary algorithm [12].

(3) Calculate the immigration rate and emigration rate for each island. Good solutions have high emigration rates and low immigration rates. Bad solutions have low emigration rates and high immigration rates.

(4) Probabilistically choose the immigrating islands based on the immigration rates. Use roulette wheel selection based on the emigration rates to select the emigrating islands.

(5) Migrate randomly selected SIVs (i.e., independent solution variables) based on the selected islands in the previous step.

(6) Probabilistically perform mutation for each island.

(7) Replace the worst islands in the population with the previous generation's elite islands.

(8) If the termination criterion is met, terminate; otherwise, go to step 3.

BBO is based on migration between islands, and each candidate solution in a BBO population is referred to as an *island*. BBO therefore uses some of the same terminology as that in island models in evolutionary computing. Island models were first introduced in distributed genetic algorithms (GAs) [13] designed for parallel computing. In island models, each island typically represents a semiisolated subpopulation of candidate solutions. Each subpopulation used a different search strategy, and candidate solutions occasionally migrate between islands. The island model is an effective tool for parallelizing evolutionary algorithms and also for preserving diversity throughout the entire population. A schema-based model of an island GA was developed and studied in [14], and a Markov-based model was proposed in [15]. Separability, population size, and convergence of island models were studied in [16]. Several parallel island EAs are reviewed and surveyed in [17]. The design and theoretical analysis of island model migration strategies are studied in [18]. Island models have been applied to practical optimization problems in several works, including [19–21].

Although BBO terminology uses the term *island*, the definition of a BBO island is different than the definition of an island in the island model. In the island model each island represents a subpopulation of candidate solutions, while in BBO each island represents a single candidate solution. We use the term *archipelago* (which is a group of islands) to represent a BBO subpopulation. In the remainder of this paper, we discuss the multiarchipelago BBO structure for optimizing complex systems. The multiarchipelago BBO structure is similar to the standard island model structure of EAs but includes a significant difference. Island model EAs are generally designed to solve a single problem. However, our multiarchipelago BBO algorithm is designed for complex systems with multiple subsystems, with each archipelago consisting of multiple islands and designed to optimize a unique subsystem.

BBO has recently become popular in both academia and industry. As with other EAs, BBO can be considered as a family of algorithms, or *metaheuristic*, and can be easily modified for application to various types of problems. BBO has been used for parameter selection in electric discharge machining [22], power flow optimization [23], robot control tuning [24], cancer classification [25], and many other applications.

Since its inception, BBO has been modified in a variety of ways. For example, it has been hybridized with differential evolution [26], ant colony optimization [27], particle swarm optimization [28], artificial bee colony optimization [29], and harmony search [30]. BBO has also been extended to special types of optimization problems, such as those with noisy fitness function evaluations [31], those with constraints [32], those with discrete search spaces [33], and those with multiple objectives [34].

2. BBO for Complex Systems

BBO was invented less than a decade ago, but according to [10] it provides competitive optimization performance with ACO, differential evolution (DE), evolutionary strategy (ES), GA, population-based incremental learning (PBIL), particle swarm optimization (PSO), stud genetic algorithm (SGA), and many other algorithms. This is the reason we extend BBO to complex systems.

The original BBO algorithm was designed for a single objective, no constraints, and single system problems. But since then BBO has been extended to multiobjective problems [35] and multiconstraint problems [36]. As we recall from Section 1, the major feature of the complex system is its multisubsystem structure. Therefore, our major goal in this paper is to extend BBO to systems with multisubsystems, where each subsystem contains multiobjectives and multiconstraints. Our new algorithm is called BBO/Complex.

Our first BBO extension involves its environment. The original BBO environment is an archipelago that consists of islands. The islands represent possible solutions to the problem. This BBO environment is based on the premise that BBO is a single system optimization algorithm. Complex systems contain more than one subsystem, each of which is partially independent from the others. Therefore, the environment of BBO/Complex includes n archipelagos, where n is the number of subsystems. The second difference between BBO and BBO/Complex involves objectives and constraints. The original BBO algorithm only includes one objective and no constraints, but BBO/Complex includes multiobjectives and multiconstraints. The new environment of BBO/Complex is as follows [37].

(1) $P = \{A\_1, A\_2, A\_3, \ldots\}$ is a population that is comprised of archipelagos. Each archipelago corresponds to one subsystem.

(2) $A\_h = \{I\_h1, I\_h2, I\_h3, \ldots; O\_h1, O\_h2, O\_h3, \ldots; C\_h1, C\_h2, C\_h3, \ldots\}$ is an archipelago that is comprised of islands I_hi, objectives O_hi, and constraints C_hi.

As previously discussed, each archipelago is an analogy for a subsystem. So each archipelago contains three groups of components. The first group of components is a group of islands, and each island is a possible solution to the subsystem optimization problem. The second group of components is a group of objectives for the subsystem. The last group of components is the set of constraints for the subsystem. The combination of all three groups of components in the subsystem is called an archipelago.

(3) $I\_hi = \{S\_hi1, S\_hi2, S\_hi3, \ldots\}$ *is* an island that is comprised of SIVs, also called candidate solution features, independent variables, or design variables, which are denoted as S_hij.

Mutation in BBO/Complex is identical to that in standard BBO. But migration in BBO/Complex needs to be modified due to the fact that the environment of BBO/Complex contains more than one subsystem. In the following sections we consider two types of migration: within-subsystem migration and cross-subsystem migration.

2.1. Within-Subsystem Migration. Within-subsystem migration contains two parts: a ranking system and a modified version of the BBO migration algorithm. In standard BBO, the fitness of an island is linearly related to the objective function because the system consists of only one objective function and no constraints. So the only performance measurement comes from the objective function. But in a complex system, the performance of an island is not reflected by only one objective function. Due to the fact that each subsystem contains multiobjectives and multiconstraints, we combine all of this information to determine the fitness of an island and its resulting migration rate. We note here that Pareto-optimal solutions are often used in multiobjective algorithms. But Pareto approaches require decision makers to select a single solution from a set of Pareto-optimal solutions, all of which are considered to be equally optimal. The Pareto approach has the advantage of providing multiple candidates to the decision maker as potential solutions but has the drawback of requiring the decision maker to select from a potentially large set of such candidate solutions. Our approach avoids the need for a human decision maker, which may be desirable for certain problems. In this paper, a ranking system is introduced for BBO/Complex which is a modified version of the nondominated ranking system (NDRS) [38].

NDRS was initially designed for single systems with multiobjectives [39]. NDRS eliminates the weighting factors used in weighted ranking algorithms. NDRS can be easily deployed in almost any optimization algorithm without major modification [40]. An updated version of NDRS was introduced in [41] as the ranking system in the multiobjective genetic algorithms (MOGA). That version uses inconsecutive integers as ranks to reflect the relative performance of each individual in a population. We are inspired here by both NDRS and the MOGA ranking system. But neither NDRS nor MOGA deal with constraint violation, which is a major concern in our work, as well as in most real-world optimization problems. So our modified NDRS considers constraint violations. We consider two factors that determine the relative performance of a candidate solution: fitness values and constraint violations. In our modified NDRS, the constraints have a higher priority than the fitness values. Violations of constraints significantly degrade the relative rank of individuals. Assume that we have a subsystem: the population size is n; the number of objectives is m; the number of constraints is k; R_i is the rank of the ith island (to be determined below); and V_i is the number of constraint violations of the ith island. Algorithm 1 outlines the modified NDRS procedure.

After performing the above version of NDRS, we have the rank of each island in the subsystem. A smaller rank means better performance. For example, suppose that we have 4 islands and each of the islands has 3 objectives and 3 constraints. The objective and constraint violation information is as in Table 1. Based on those, the rank of each island is calculated according to the modified NDRS method in Table 1.

The ranks obtained from the modified NDRS are shown in Table 1, but one thing that needs to be mentioned is that the ranks assigned to the islands are 0, 4, 5, and 9 rather than 0, 1, 2, and 3. Ranks are not necessarily consecutive integers. The reason is that NDRS reflects the performance of an island by including the number of partial domination counts in a rank rather than simply ordering the islands. This gives more granularity for rank values, which is important when statistically choosing migrating islands in BBO.

2.2. Cross-Subsystem Migration.

2.2. Cross-Subsystem Migration. Standard BBO only contains one type of migration, within-subsystem migration, which has been modified for BBO/Complex as shown before. But BBO/Complex also includes cross-subsystem migration. Cross-subsystem migration is different because each subsystem has its own ranking system. The comparison of ranks across subsystems is meaningless, because ranks assigned to each island in a subsystem only represents the relative goodness of the island in that specific subsystem. If we consider two islands from two different subsystems, we cannot determine which island is better by simply comparing their ranks, because ranks from different subsystems are calculated differently based on the different subsystem objectives and constraints. Instead, cross-subsystem migration is based on three factors—distance between islands, the similarity level of objectives, and the similarity level of constraints.

2.2.1. Distance between Islands. The first factor to consider in cross-subsystem migration is the distance between islands. As we know, heuristic algorithms require population diversity [12]. BBO migration is based on sharing SIVs among islands. If the population has a low diversity, most of the islands are similar to each other, and the probability that an island improves after migration is low. In this case, migration may not effectively contribute to improvement in the population.

Mutation is the technique that introduces new SIVs to the population, and mutation does not depend on the diversity of the population. But the mutation rate is usually a small number, for example, 1%, because large mutation rates negate the effectiveness of migration and reduce the evolutionary algorithm to a random search. The new information introduced to the population through mutation sometimes includes useful SIVs. But most of the time, those SIVs are useless and can even degrade the population. Mutation, in general, is not a rapid or efficient technique for evolution.

Usually we use the Euclidean distance to calculate the differences between islands. This calculation is straightforward for islands with the same structure. The Euclidean distance between islands a and b in archipelago h, both of which have c SIVs, is

$$D_{\mathrm{hab}} = \sqrt{\sum_{k=1}^{c} \left(S_{\mathrm{hak}} - S_{\mathrm{hbk}}\right)^2}. \quad (1)$$

This calculation is valid if and only if both islands share the same structure, which means they have the same SIV type at the same location. But in a complex system, subsystems usually have different island structures. That is, the independent variables in subsystems are not commensurate. For example, the SIV types in island 1 may be labeled types 1, 2, and 3, but the SIV types in island 2 may be labeled 2, 3, and 4. Equation (1) is not appropriate to calculate the distance between islands 1 and 2, because we cannot find the corresponding SIVs on both islands for the type 1 SIV and the type 4 SIV.

For BBO/Complex, we need a new technique to calculate the distance between islands with different structures. The partial distance strategy (PDS) is widely used in statistics to calculate Euclidean distances with missing data [42]. This is similar to our situation. Instead of missing data, we have missing SIV types. In order to implement PDS, we need to modify the data structure of the islands. First, we define each island to include all the SIV types on all islands. If an island did not originally include a specific SIV type, we assign

```
$R_1 = R_2 = \cdots = R_n = 0$;
$V_1 = V_2 = \cdots = V_n = 0$;
for $i = 1$ to $n$ do
        for $c = 1$ to $k$ do
                if constraint $c$ of island $i$ is violated then
                        $V_i = V_i + 1$
                end if
        end for
end for
for $i_1 = 1$ to $n$ do
        for $i_2 = i_1$ to $n$ do
                if $V_{i1} > V_{i2}$
                        $R_{i1} = R_{i1} + m$
                else if $V_{i1} < V_{i2}$
                        $R_{i2} = R_{i2} + m$
                else if $V_{i1} = V_{i2}$
                        for $o_1 = 1$ to $m$ do
                                if objective $o_1$ of island $i_1$ is better than $o_1$ of $i_2$ then
                                        $R_{i2} = R_{i2} + 1$
                                else if objective $o_1$ of island $t_2$ is better than $o_1$ of $t_1$ then
                                        $R_{i1} = R_{i1} + 1$
                                end if
                        end for
                end if
        end for
end for
```

Algorithm 1: Modified non-dominated ranking system (NDRS). V_i is the number of constraint violations of the ith island, and R_i is the relative rank of the ith island, where a lower rank is better. m is the number of optimization objectives.

Table 1: Rank calculation example with the modified NDRS. A lower objective means better performance, and lower ranks are better than higher ranks.

	Objective 1	Objective 2	Objective 3	Constraint violation	Rank
Island 1	1	2	3	0	0
Island 2	2	4	2	1	4
Island 3	3	1	4	1	5
Island 4	1	1	1	2	9

an *N/A* value to the SIV and treat it as missing data. The implementation of PDS in BBO/Complex is given as follows:

$$D_{\text{ghab}} = \begin{cases} \dfrac{t}{K_{\text{ghab}}} \sqrt{\displaystyle\sum_{k=l}^{t} \left(S_{\text{gak}} - S_{\text{hbk}}\right)^2 K_{\text{ghabk}}}, & \text{if } K_{\text{ghab}} > 0, \\ 0, & \text{if } K_{\text{ghab}} = 0, \end{cases}$$

$$K_{\text{ghabk}} = \begin{cases} 0, & \text{if } S_{\text{gak}} = \dfrac{N}{A} \text{ or } S_{\text{hbk}} = \dfrac{N}{A}, \\ 1, & \text{if } S_{\text{gak}} \neq \dfrac{N}{A}, \text{ and } S_{\text{hbk}} \neq \dfrac{N}{A}, \end{cases}$$

$$K_{\text{ghab}} = \sum_{k=1}^{t} K_{\text{ghabk}}. \tag{2}$$

D_{ghab} is the partial distance between island a in archipelago g and island b in archipelago h, and t is the total number of SIV types. As an example, suppose we have 2 islands: island 1 = $[0, 1, 2, 3, N/A, 4]$ and island 2 = $[1, 3, N/A, N/A, 5, 5]$. Island 1 has 5 SIVs and island 2 has 4 SIVs, and the two islands have 3 SIVs in common. Then the distance is calculated based on (2) as $D_{12} = 4.90$.

2.2.2. Similarities between Objectives and Constraints. The second and third factors in the island distance calculation are the similarity level of the objectives and the similarity level of the constraints. Subsystems with similar objectives and constraints are more likely to benefit each other through migration than subsystems that are not closely related. Our calculation of the similarity level is based on the fast similarity level calculation (FSLC) [37]. Suppose there are two islands, each of which has a vector of variables: $U = [u_1, u_2, u_3, \ldots]$ and $V = [v_1, v_2, v_3, \ldots]$ (either objectives or constraints). The similarity level (SL) of these vectors is calculated by FSLC in Algorithm 2.

```
SL = 0;
**for** each $u \in U$
    **for** each $v \in V$
        **if** $u$ and $v$ are the same type **then**
            SL = SL + 1
        **end if**
    **end for**
**end for**
```

Algorithm 2: Similarity level calculation. U and V are the sets of objectives or constraints of two islands (candidate solutions).

2.2.3. Summary of Cross-Subsystem Migration. Now that we have discussed the three factors for cross-subsystem migration, we summarize cross-subsystem migration as follows. First, calculate the migration probability between islands based on the similarity level between subsystems:

$$P_{\text{migration}} = \begin{cases} \dfrac{1}{2}\left(\dfrac{\text{OS}}{\text{OS}_{\max}} + \dfrac{\text{CS}}{\text{CS}_{\max}}\right), & \\ \qquad \text{if } \text{OS}_{\max} > 0,\ \text{CS}_{\max} > 0, & \\ \dfrac{1}{2}\dfrac{\text{OS}}{\text{OS}_{\max}}, & \\ \qquad \text{if } \text{OS}_{\max} > 0,\ \text{CS}_{\max} = 0, & \\ \dfrac{1}{2}\dfrac{\text{CS}}{\text{CS}_{\max}}, & \\ \qquad \text{if } \text{OS}_{\max} = 0,\ \text{CS}_{\max} > 0, & \\ 0, & \\ \qquad \text{if } \text{OS}_{\max} = 0,\ \text{CS}_{\max} = 0, & \end{cases} \tag{3}$$

OS is objective similarity level between two islands,

$\text{OS}_{\max}$ is the maximum interarchipelago objective similarity level in the population,

CS is constraint similarity level between two islands,

$\text{CS}_{\max}$ is the maximum inter-archipelago constraint similarity level in the population.

The probability for a pair of subsystems to perform cross-subsystem migration is linearly related to the above migration probability. After that, we need to choose emigrating islands for each immigrating island. We use roulette wheel [43] to select the emigrating island. Islands with better partial distances will have better chance to be selected as the emigrating island. Figure 4 shows an example of emigrating island selection across subsystems.

2.3. Summary of BBO/Complex. BBO/Complex is summarized as follows.

(1) Define the control parameters: population size, stopping criteria, mutation probability, and elitism parameter. For example, a typical setup for BBO is that population size is 100, stopping criteria is 100,000 cost function calls, mutation probability is 0.05, and elitism parameter is 1.

(2) Initialize the population. This is usually done with randomlygenerated individuals.

(3) Calculate the constraint and objective similarity levels between all pairs of subsystems.

(4) Calculate the rank of islands in each subsystem.

(5) Perform within-subsystem migration: probabilistically choose the immigrating islands based on the island ranks. Use roulette wheel selection based on the emigration rates to select the emigrating islands. Emigration rates are linearly related to the island ranks. After each immigrating island selects its corresponding emigrating island, we perform within-subsystem migration. Each SIV in an immigrating island will have a chance to be replaced by an SIV from an emigrating island.

(6) Perform cross-subsystem migration: find suitable pairs of subsystems based on similarity levels. Calculate distances between each pair of islands from different subsystems. Use roulette wheel selection based on partial distances to select the emigrating islands. Then, we begin cross-subsystem migration. Each SIV in an immigrating island will have a chance to be replaced by a SIV from an emigrating island, and this probability is $P_{\text{SIVmigration}}$ which can be predefined by users.

(7) Probabilistically perform mutation on each island based on the mutation probability.

(8) Save the islands in each subsystem with best performances as elite islands. Replace the worst islands in the population with the previous generation's elite islands.

(9) If the termination criterion is not met, go to step 4; otherwise, terminate.

The structure of BBO/Complex is conceptually different than MDF, IDF, and CO. As we see from Figures 1, 2, and 3, that of MDF, IDF, and CO provide different strategies to optimize systems. But they are just frameworks, and we can choose any optimization method, like gradient descent or a genetic algorithm (GA), as the optimizer within the framework. But BBO/Complex is in a different category, because it includes both the framework and the optimization algorithm, as shown in Figure 5. It provides an efficient way to communicate between subsystems and provides a

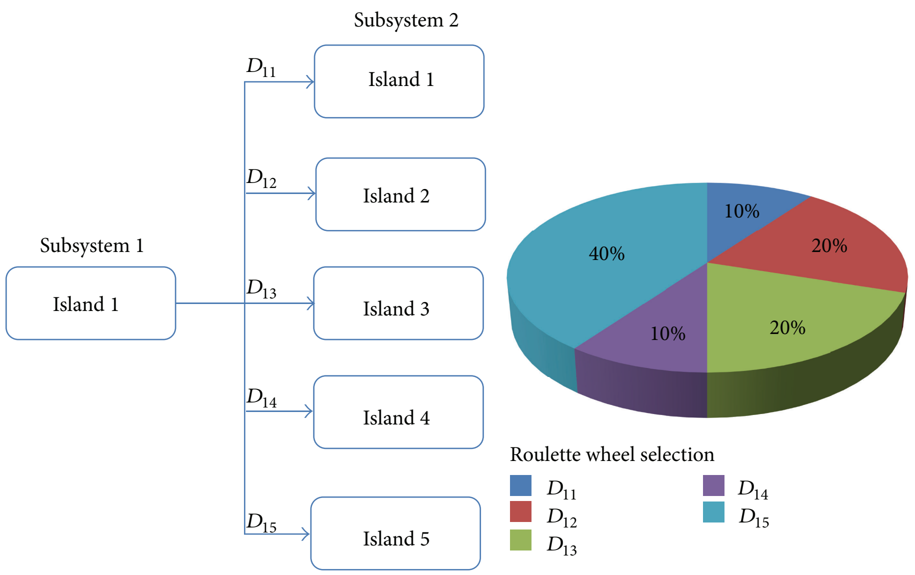

Figure 4: An example of emigrating island selection for immigration to island 1 in subsystem 1. First, calculate the partial distances between island 1 in subsystem 1 and each island in subsystem 2. Then create a roulette wheel based on the partial distances. Finally, probabilistically select the emigrating island based on the roulette wheel.

unique migration strategy to share information both within and across subsystems. Comparing Figures 1, 2, 3, and 5, we see that cross-subsystem migration in BBO/Complex is an innovation that can significantly enrich communication among subsystems compared to more traditional MDO methods.

3. Simulation Results

In this section, we compare the performance of BBO/Complex in real-world benchmark problems with other well-known MDO algorithms: MDF, IDF, and CO. As we mentioned before, these three MDO algorithms are frameworks which require an additional optimization method as a complementary but essential component. The optimization algorithm we use in all three of these MDO algorithms is BBO without cross-subsystem migration. The benchmark problems are obtained from [44] and include the speed reducer problem, the propane combustion problem, the heart dipole problem, and the power converter problem. Each benchmark contains several subsystems, and each subsystem contains multiobjectives and multiconstraints. Detailed information about each benchmark can be found in the Appendix.

The reason we choose these benchmarks is that they can be formulated as a complex system with interconnected subsystems. There are two decomposition strategies: one is based on the physical system and one is based on the system requirements. In this paper we decompose the systems based on system requirements. Based on [45, 46], traditional MDO algorithms usually lack the capability of dealing with multiobjectives, so their decomposition is based on the principle that each subsystem has one objective and multiconstraints. This type of decomposition is suitable for traditional optimization methods because it avoids the need to consider all objectives at once. Due to the fact that BBO belongs to the heuristic algorithm category and with supporting results from [34, 47, 48], BBO/Complex is expected to perform well on multiobjective problems. It has more flexible decomposition options compared to traditional MDO algorithms.

Our decomposition option for BBO/Complex is that each subsystem has multiobjectives and multiconstraints. But in order to provide a fair comparison between other MDO algorithms and BBO/Complex, we also introduce a BBO/Complex version that uses the same decomposition strategy as the other MDO algorithms. So we have two versions of BBO/Complex in this section: the first one uses the same decomposition method as CO, MDF, and IDF and is called BBO/Complex/Single; the other one uses multiobjectives in each subsystem and is called BBO/Complex/Multi.

For each benchmark test, we compare the performance of each algorithm using both feasibility and cost. We perform 100 Monte Carlo simulations for each algorithm and each benchmark problem to accurately measure performance. The termination criterion is 100,000 cost function evaluations. The feasibility index is calculated for each generation as the average number of constraint violations among all Monte Carlo simulations. The feasibility index is 0 if there are no violations. The second performance metric is based on the cost function values. We calculate the average cost values among all Monte Carlo simulations and then use the modified NDRS to obtain cost rank values. The optimization goal for each benchmark is to find the minimum value of the cost without violating any constraints. Since each benchmark contains multiobjectives, we use NDRS to calculate the rank for each algorithm based on its cost. But we have two priority levels: the first goal is to find feasible solutions; the second goal is to reduce cost. Priority level one overrides priority level two.

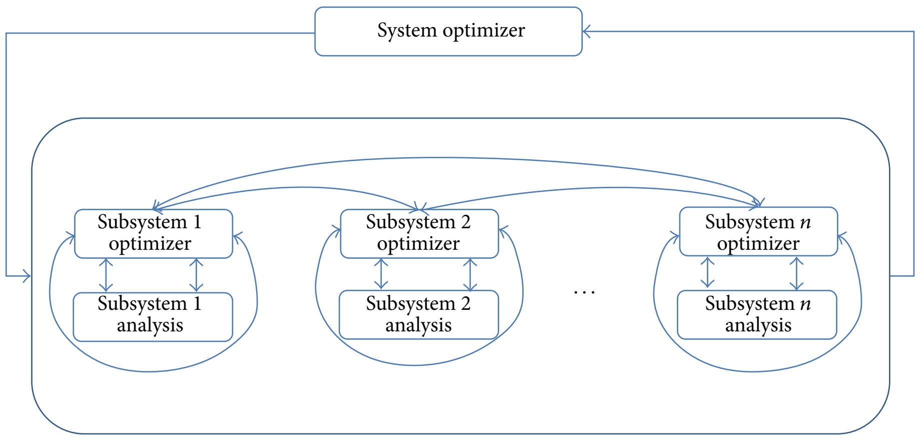

Figure 5: BBO/Complex formulation.

Table 2: NDRS cost rank and feasibility for the speed reducer problem after 100,000 function calls. For each metric, a smaller number means better performance.

Algorithm	NDRS cost rank	Feasibility
BBO/Complex/Single	4	0.27
MDF	5	0.60
CO	6	0.86
BBO/Complex/Multi	7	1.57
IDF	8	3.00

3.1. The Speed Reducer Problem. The first benchmark we test is the speed reducer problem. It contains 3 objectives, 11 constraints, and 7 design variables, as detailed in the Appendix. The performance of all algorithms in the first benchmark is shown in Table 2 and Figure 6, which show that BBO/Complex/Single has the best performance in the speed reducer benchmark, including the best cost rank and the best feasibility level. MDF, CO, and BBO/Complex/Multi are slightly worse than BBO/Complex/Single. IDF has the worst performance in terms of both cost rank and feasibility level. Note that both feasibility and cost at the beginning of the simulation start at different values for different algorithms. This is because when we initially evaluate the performance of algorithms, they have already been optimized at the subsystem level.

3.2. The Power Converter Problem. The second benchmark is the power converter problem. It has 6 design variables, 8 state variables, 2 objectives, and 4 constraints, as detailed in the Appendix. Table 3 and Figure 7 show the performance of the algorithms in the power converter problem. The performances of all algorithms are fairly close to each other. We have good results in this problem because all algorithms achieve a 0 feasibility level. MDF is the best algorithm in terms of cost, and BBO/Complex/Multi has the second best performance.

3.3. The Heart Dipole Problem. The third benchmark is the heart dipole problem. It has 6 design variables, 2 objectives, and 5 constraints, as detailed in the Appendix. Table 4 and Figure 8 show that BBO/Complex/Single and BBO/Complex/Multi are the only algorithms which achieve a 0 feasibility level, which means that the best individuals for each Monte Carlo run are feasible. When we combine cost rank and feasibility, BBO/Complex/Multi has the best performance in this benchmark, and BBO/Complex/Single is the second best.

3.4. The Propane Combustion Problem. The fourth benchmark is the propane combustion problem. It has 1 design variable, 3 objectives, and 4 constraints, as detailed in the Appendix. According to Table 5 and Figure 9, BBO/Complex/Multi is the best algorithm for this benchmark because it is the only algorithm that achieves a 0 feasibility level. BBO/Complex/Single achieves the second best performance with a feasibility level slightly greater than 0.

3.5. Summary of Benchmark Tests. The benchmark results show that BBO/Complex/Multi is the only algorithm that obtains feasible solutions in three of the benchmarks. For the speed reducer benchmark, none of the algorithms finds a feasible solution, but BBO/Complex/Single comes the closest. Among all four benchmarks, BBO/Complex/Multi achieves the best performance twice and the second best performance once, and BBO/Complex/Single achieves the best performance once and the second best performance twice. Among the non-BBO/Complex algorithms, MDF is the best, achieving the best performance once and the second best performance once.

4. Conclusion

Heuristic algorithms are powerful and proven optimization techniques whose structures are motivated by nature. In contrast with more traditional optimization methods, heuristic

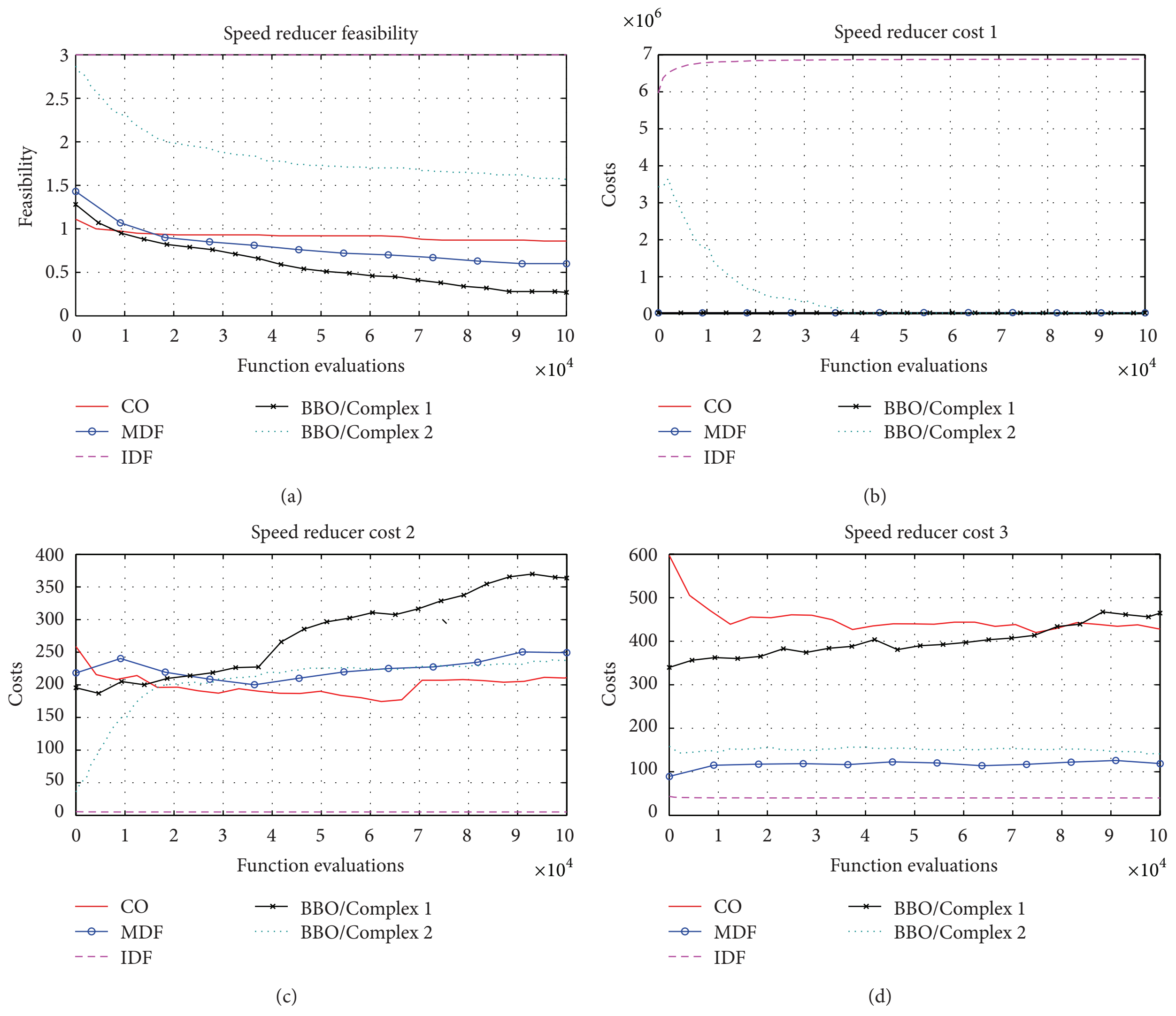

Figure 6: The feasibility and cost of each objective for the speed reducer problem.

algorithms are intuitive and easy to apply, even to systems with complex structures.

Complex systems include multisubsystems, multiobjectives, and multiconstraints and have been deployed in every type of modern industry. But optimization methods for complex systems are lagging behind the implementation of complex systems in industry. CO, MDF, and IDF are often applied to the optimization of complex systems. Those methods have made many contributions to modern industry, but this paper has aimed to take complex systems optimization to the next level.

In this paper, BBO, a newly developed heuristic algorithm, has been extended and applied to complex system optimization. Our new algorithm is called BBO/Complex. BBO/Complex uses the original framework of standard BBO but extends it to a multiarchipelago environment to suit the structure of complex systems. BBO/Complex has one significant difference from its predecessors—it combines the optimization framework and the low-level optimization approach into a single algorithm. This is quite different from MDF, IDF, and CO, all of which are only frameworks for complex systems and need a low-level optimization method as an additional tuning parameter. The low-level optimization approaches incorporated in MDF, IDF, and CO are typically traditional algorithms like gradient descent and Newton's method. But those algorithms can easily get stuck

Table 3: NDRS cost rank and feasibility for the power converter problem after 100,000 function calls. For each metric, a smaller number means better performance.

Algorithm	NDRS cost rank	Feasibility
MDF	1	0
BBO/Complex/Multi	3	0
IDF	4	0
BBO/Complex/Single	6	0
CO	6	0

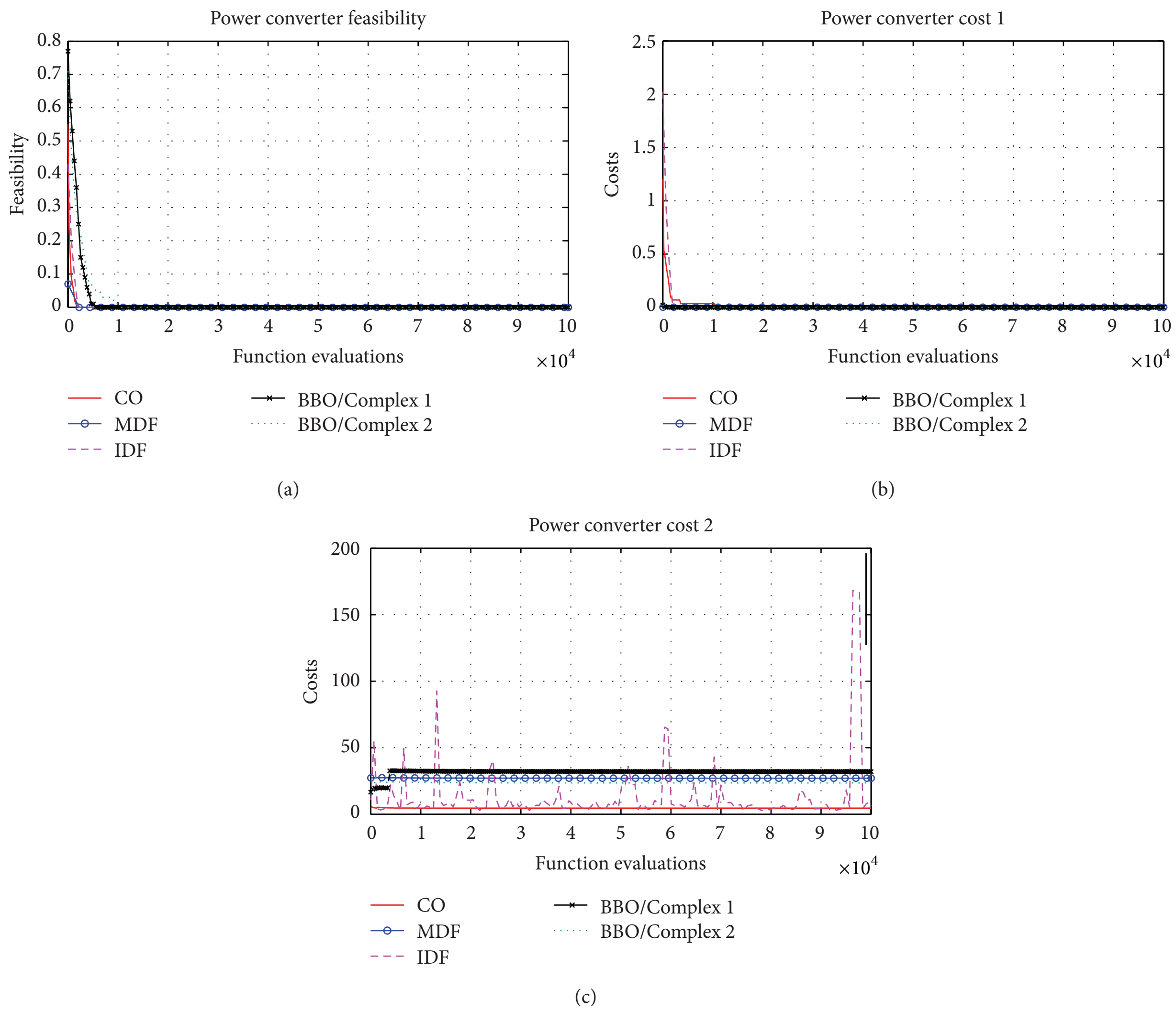

FIGURE 7: The feasibility and cost of each objective for the power converter problem.

in a local optimum. Based on [49, 50], standard BBO can guarantee convergence to the optimal solution given enough generations. Besides the traditional advantages of BBO, the BBO/Complex algorithm also introduces new features, like a ranking system that evaluates candidate solutions based on both performance and constraints, the use of a partial distance strategy to maintain the diversity of the population, within-subsystem migration for information sharing within subpopulations, and cross-subsystem migration for information sharing between subpopulations. The simulation results indicate that BBO/Complex is a competitive multidisciplinary optimization algorithm.

Future work for BBO/Complex can be extended in two directions: speed and adaptation. Convergence speed is one of the primary concerns for heuristic algorithms. Parallel computation can be used to decrease convergence time by dividing a task into multiple subtasks and solving them in parallel. One of the classic parallel computation models is the master-slave model. The master is in charge of job assignment and global calculations. The slaves perform subtasks that are assigned by the master and return the results to the master. This structure can be adapted to BBO/Complex by viewing the master as the system optimizer and each slave as a subsystem optimizer. Computation time can be decreased

TABLE 4: NDRS cost rank and feasibility for the heart dipole problem after 100,000 function calls. For each metric, a smaller number means better performance.

Algorithm	NDRS cost rank	Feasibility
BBO/Complex/Multi	2	0
BBO/Complex/Single	6	0
MDF	6	0.03
IDF	4	1.00
CO	2	2.00

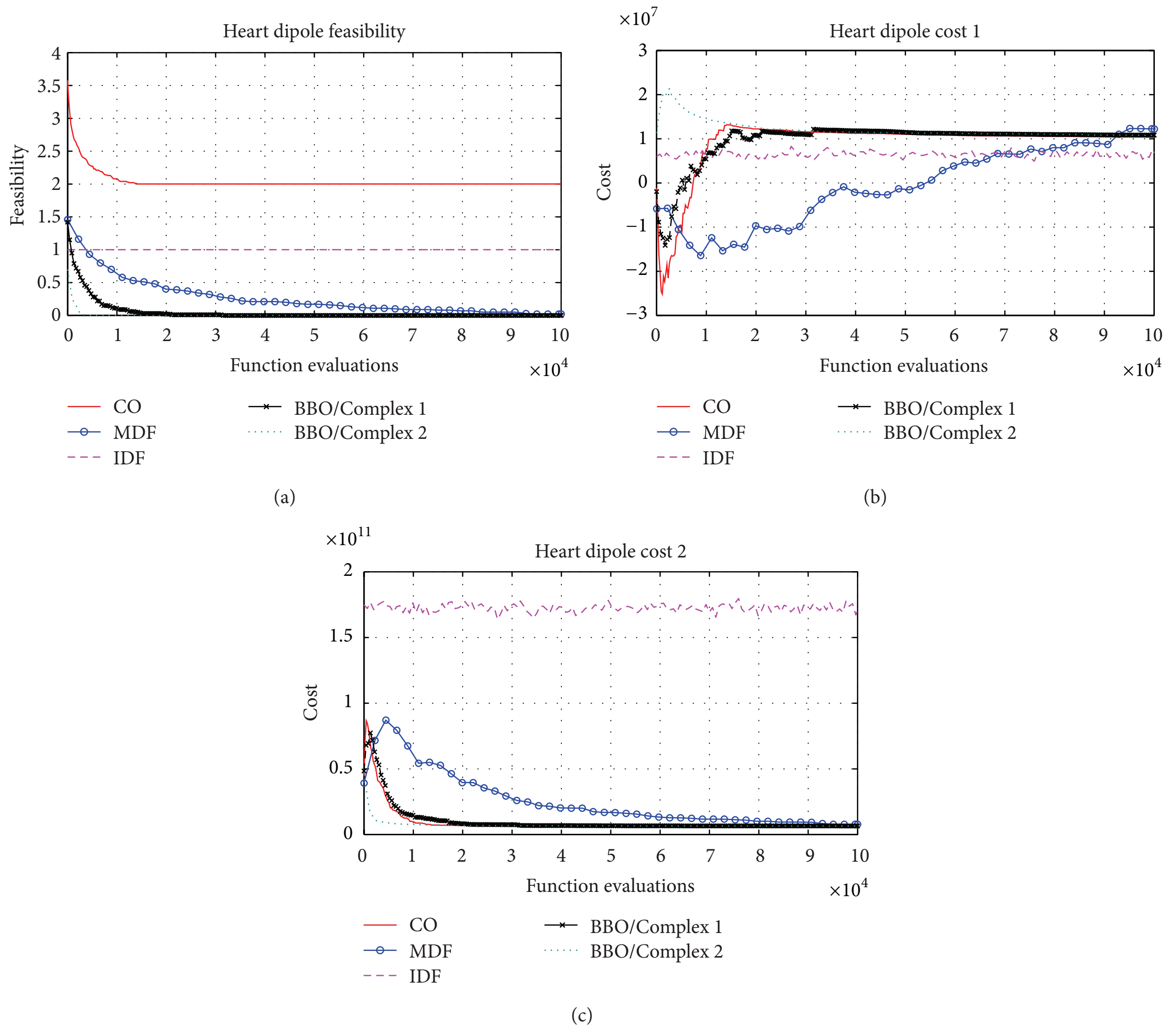

Figure 8: The feasibility and cost of each objective for the heart dipole problem.

dramatically with this structure, especially for problems with a large number of subsystems.

The second direction for future research in BBO/Complex is adaptation. In BBO/Complex, we find a solution to a complex system with a combination of within-subsystem migration and cross-subsystem migration. But other types of migration could be implemented. A proper migration method can significantly increase performance for different types of problems. So we can design a series of migration methods, like migration for complex systems with tight subsystem coupling, migration for complex systems with loose subsystem coupling, and migration for complex systems with many design variables. Then we can classify the migration methods according to their performances in various types of problems and create a BBO/Complex algorithm that adaptively chooses the most efficient migration methods according to the selected problem.

Table 5: NDRS cost rank and feasibility for the propane combustion problem after 100,000 function calls. For each metric, a smaller number means better performance.

Algorithm	NDRS cost rank	Feasibility
BBO/Complex/Multi	7	0
BBO/Complex/Single	5	0.08
CO	7	0.16
MDF	3	1.00
IDF	5	1.00

Appendix

This appendix gives details about the benchmark problems used in this paper.

Speed Reducer (see [44, 45, 51]). The speed reducer problem is a gear box design problem. The objective is to minimize

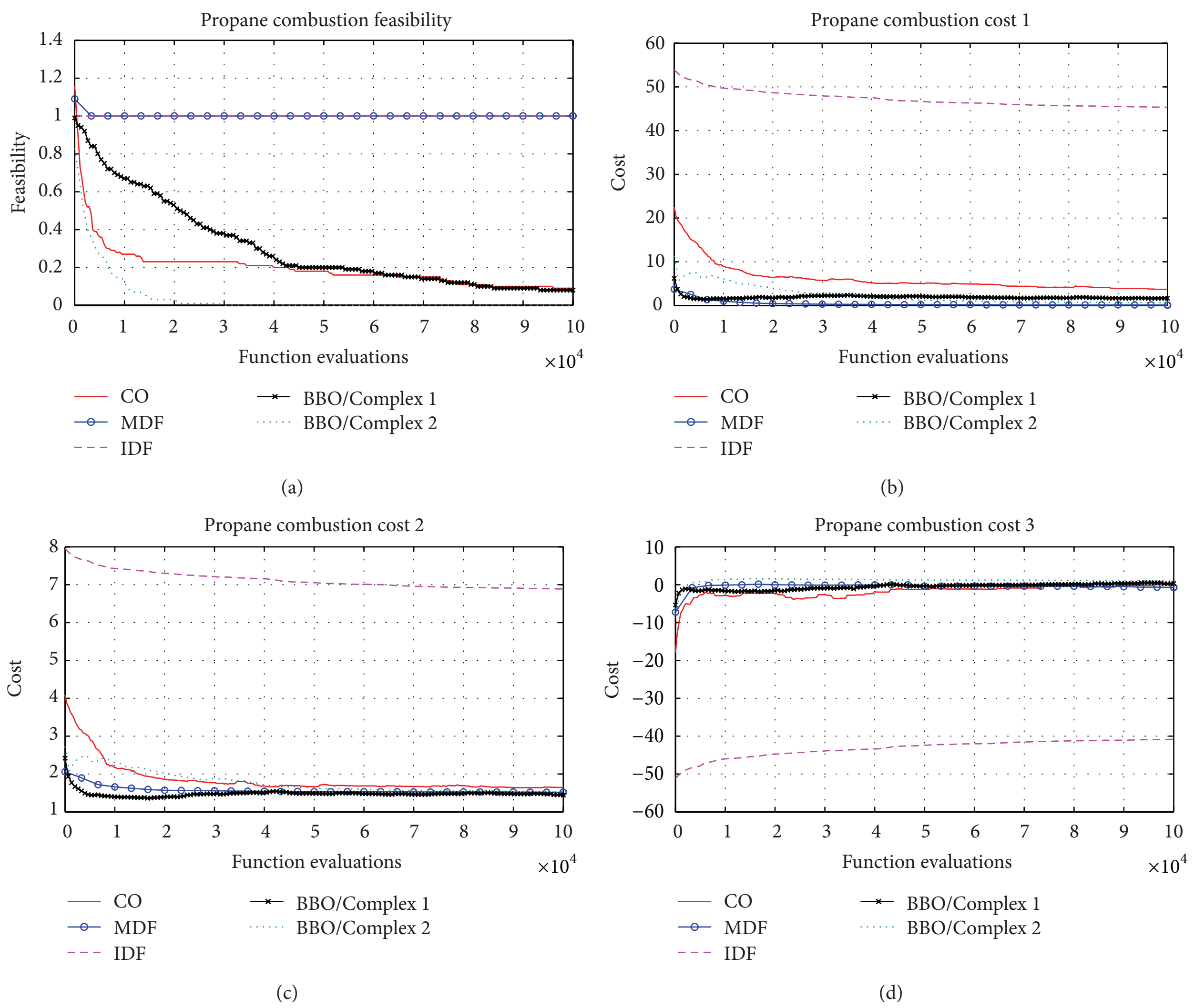

FIGURE 9: The feasibility and cost of each objective for the propane combustion problem.

the gear box weight and the von Mises stresses for shafts 1 and 2. This problem contains 3 objectives, 11 constraints, and 7 design variables. This problem is defined as follows:

$$\begin{aligned} \min F_1 &= 0.7854 x_1 x_2^2 \left(3.3333 x_3^2 + 14.9334 x_3 - 43.0934\right) \\ &\quad - 1.5079 x_1 \left(x_6^2 + x_7^2\right) + 7.477 \left(x_6^3 + x_7^3\right) \\ &\quad + 0.7854 \left(x_4 x_6^2 + x_5 x_7^2\right), \\ \min F_2 &= \sqrt{\left(\frac{745 x_4}{x_2 x_3}\right)^2 + 1.69 \times 10^7}, \\ \min F_3 &= \sqrt{\left(\frac{745 x_5}{x_2 x_3}\right)^2 + 1.575 \times 10^8}, \end{aligned} \tag{A.1}$$

such that the following constraints hold:

$$\begin{aligned} g_1 &= \frac{27}{x_1 x_2^2 x_3} - 1 \le 0, \\ g_2 &= \frac{397.5}{x_1 x_2^2 x_3^2} - 1 \le 0, \\ g_3 &= \frac{1.93 x_4^3}{x_2 x_3 x_6^4} - 1 \le 0, \\ g_4 &= \frac{1.93 x_5^3}{x_2 x_3 x_7^4} - 1 \le 0, \\ g_5 &= \frac{\sqrt{(745 x_4 / x_2 x_3) + 1.69 \times 10^7}}{0.1 x_6^3} - 1100 \le 0, \\ g_6 &= \frac{\sqrt{(745 x_5 / x_2 x_3) + 1.575 \times 10^8}}{0.1 x_6^3} - 850 \le 0, \end{aligned}$$

$$g_7 = x_2x_3 - 40 \leq 0,$$
$$g_8 = \frac{x_1}{x_2} - 12 \leq 0,$$
$$g_9 = \frac{-x_1}{x_2} + 4 \leq 0,$$
$$g_{10} = \frac{1.5x_6 + 1.9}{x_4} - 1 \leq 0,$$
$$g_{11} = \frac{1.1x_7 + 1.9}{x_5} - 1 \leq 0. \qquad \text{(A.2)}$$

The objectives, decision variables, and constraints are defined as follows:

F_1: overall weight of gear box,
F_2: von Mises stress for shaft 1,
F_3: von Mises stress for shaft 2,
x_1: gear face width,
x_2: tooth module,
x_3: number of teeth of pinion,
x_4: distance between bearing 1,
x_5: distance between bearing 2,
x_6: diameter of shaft 1,
x_7: diameter of shaft 2,
g_1: bending stress of gear tooth,
g_2: contact stress of gear tooth,
g_3: transverse deflection of shaft 1,
g_4: transverse deflection of shaft 2,
g_5: stress in shaft 1,
g_6: stress in shaft 2,
g_7–g_{11}: dimension requirement for shafts.

Power Converter (see [44, 51]). The power converter problem consists of two subsystems—the electrical subsystem and the loss subsystem. It has 6 design variables, 8 state variables, 2 objectives, and 4 constraints. The system is described as follows:

$$\begin{aligned}\min F_1 &= \left|0.78 \times 10^4 {x_1}^2 \left(6x_6 + \frac{\pi x_1}{2}\right)\right| \\ &\quad + \left|6.747 \times 10^4 x_1x_2x_3\right|, \qquad \text{(A.3)} \\ \min F_2 &= \left|25x_5\right| + \left|\frac{5 \times 10^2 (1 - y_2)}{88y_2}\right|,\end{aligned}$$

such that the following constraints hold:

$$g_1 = \frac{2x_2 \left(x_2 - 2 \times 10^{-3} - x_2x_3\right)}{0.4} \geq 0,$$
$$g_2 = \frac{5 \times 10^2 - \left(5.65 (1 - y_3)/10^5 x_4\right)\left(0.3 \times 10^{-4}/x_5\right)}{5} \geq 0,$$
$$g_3 = 0.3 - \frac{x_4 \left(100 + \left(5.65 (1 - y_4)\, 0.5/10^5 x_4\right)\right)}{x_2y_6} \geq 0,$$
$$g_4 = x_4 - \frac{28.25 (1 - y_4)}{10^7} \geq 0. \qquad \text{(A.4)}$$

The state variables are defined as follows:

$$\begin{aligned}y_1 &= \left|0.78 \times 10^4 {x_1}^2 \left(6x_6 + \frac{\pi x_1}{2}\right)\right| \\ &\quad + \left|6.747 \times 10^4 x_1x_2x_3\right| + \left|25x_5\right| \\ &\quad + \left|\frac{5 \times 10^2 (1 - y_2)}{88y_2}\right|, \\ y_2 &= \frac{500}{y_3 \left(\left(3.25 \times 10^2\right)/32\right)}, \\ y_3 &= \frac{500}{y_2 \left(\left(3.25 \times 10^2\right)/32\right)}, \\ y_4 &= \frac{500}{y_2 \left(\left(4.25 \times 10^2\right)/32\right)}, \\ y_5 &= \frac{7.6x_1x_2 1.724 \times 10^{-8}}{x_3}, \\ y_6 &= {x_1}^2, \\ y_7 &= \frac{\pi x_1}{2}, \\ y_8 &= \frac{5.65 (1 + y_3)}{y_6x_2 10^5}.\end{aligned} \qquad \text{(A.5)}$$

The objectives, decision variables, states, and constraints are defined as follows:

F_1: weight of primary winding,
F_2: weight of secondary winding,
x_1: core center leg width,
x_2: turns,
x_3: copper size,
x_4: inductance,
x_5: capacitance,
x_6: core window width,
y_1: component weight,

y_2: circuit efficiency,

y_3: duty cycle,

y_4: minimum duty cycle,

y_5: inductor resistance,

y_6: core cross-sectional area,

y_7: magnetic path length,

y_8: inductor value,

g_1: fill window constraint,

g_2: ripple specification,

g_3: core saturation,

g_4: minimum inductor size.

Heart Dipole (see [44, 51, 52]). The heart dipole problem is based on the electrolytic determination of the dipole moment in the heart. This problem contains 2 objectives, 5 constraints, and 6 design variables. This problem was modified from its original formulation in order to be testable with MDO algorithms. Therefore, although the problem is a common MDO benchmark, the objectives do not have any physical meaning. The problem is defined as follows:

$$\begin{aligned}
\min F_1 = {} & x_1\left((1-x_2)^2 - x_3^2\right) - 2x_1(1-x_2)x_3 \\
& + (1-x_1)\left(x_2^2 - x_4^2\right) - 2(1-x_1)x_2x_4 - 1 \\
& + x_1\left((1-x_2)^2 - x_3^2\right) + 2x_1(1-x_2)x_3 \\
& + (1-x_1)\left(x_2^2 - x_4^2\right) + 2(1-x_1)x_2x_4 - 1, \\
\min F_2 = {} & x_1(1-x_2)\left((1-x_2)^2 - 3x_3^2\right) \\
& + x_1x_3\left(x_3^2 - 3\left(1-x_2^2\right)\right) \\
& + (1-x_1)x_2\left(x_2^2 - 3x_4^2\right) \\
& + (1-x_1)x_4\left(x_4^2 - 3x_2^2\right) - 1 \\
& + x_1(1-x_2)\left((1-x_2)^2 - 3x_3^2\right) \\
& - x_1x_3\left(x_3^2 - 3(1-x_2)^2\right) \\
& + (1-x_1)x_2\left(x_2^2 - 3x_4^2\right) \\
& - (1-x_1)x_4\left(x_4^2 - x_2^2\right) - 1,
\end{aligned} \tag{A.6}$$

such that the following constraints hold:

$$\begin{aligned}
g_1 &= \left|x_3x_1 + x_4(1-x_1) - x_5(1-x_2) - x_6x_2 - 1\right| < 0.1, \\
g_2 &= \left|x_5x_1 + x_6(1-x_1) + x_3(1-x_2) + x_4x_2 - 1\right| < 0.1, \\
g_3 &= x_1\left((1-x_2)^2 - x_3^2\right) - 2x_1(1-x_2)x_3 \\
&\quad + (1-x_1)\left(x_2^2 - x_4^2\right) - 2(1-x_1)x_2x_4 - 1 > 0, \\
g_4 &= x_1\left((1-x_2)^2 - x_3^2\right) + 2x_1(1-x_2)x_3 \\
&\quad + (1-x_1)\left(x_2^2 - x_4^2\right) + 2(1-x_1)x_2x_4 - 1 > 0, \\
g_5 &= x_1(1-x_2)\left((1-x_2)^2 - 3x_3^2\right) \\
&\quad + x_1x_3\left(x_3^2 - 3(1-x_2)^2\right) \\
&\quad + (1-x_1)x_2\left(x_2^2 - 3x_4^2\right) \\
&\quad + (1-x_1)x_4\left(x_4^2 - 3x_2^2\right) - 1 > 0, \\
g_6 &= x_1(1-x_2)\left((1-x_2)^2 - 3x_3^2\right) \\
&\quad - x_1x_3\left(x_3^2 - 3(1-x_2)^2\right) + (1-x_1)x_2\left(x_2^2 - 3x_4^2\right) \\
&\quad - (1-x_1)x_4\left(x_4^2 - 3x_2^2\right) - 1 > 0.
\end{aligned} \tag{A.7}$$

The objectives, decision variables, and constraints are defined as follows:

F_1: sum of g_3 and g_4,

F_2: sum of g_5 and g_6,

x_1: magnitude of dipole 1 on x-axis,

x_2: magnitude of dipole 2 on x-axis,

x_3: magnitude of dipole 1 on y-axis,

x_4: magnitude of dipole 2 on y-axis,

x_5: coordinate of dipole 1 on x-axis,

x_6: coordinate of dipole 2 on x-axis,

x_7: coordinate of dipole 1 on y-axis,

x_8: coordinate of dipole 2 on y-axis,

g_1–g_6: predefined constraints to determine the magnitude, directions, and locations of two dipoles.

Propane Combustion (see [44, 51, 53]). The propane combustion problem is a chemical equilibrium problem. This problem contains 3 objectives, 4 constraints, and 11 design variables. This problem is described as follows:

$$\min F_1 = 2x_1 + x_2 + x_4 + x_7 + x_8 + x_9 + 2x_{10} - 10,$$
$$\min F_2 = \sqrt{x_2 x_4} - x_6\sqrt{\frac{40x_1}{x_{11}}}, \qquad x_{11} = \sum_{i=1}^{i=10} x_i,$$
$$\min F_3 = \sqrt{x_1 x_2} - x_7\sqrt{\frac{40x_4}{x_{11}}} + x_1\sqrt{x_3} - x_4 x_9\sqrt{\frac{40}{x_{11}}}, \tag{A.8}$$

such that the following constraints hold:

$$g_1 = 2x_1 + x_2 + x_4 + x_7 + x_8 + x_9 + 2x_{10} - 10 > 0,$$
$$g_2 = \sqrt{x_2 x_4} - x_6\sqrt{\frac{40x_1}{x_{11}}} > 0,$$
$$g_3 = \sqrt{x_1 x_2} - x_7\sqrt{\frac{40x_4}{x_{11}}} > 0, \tag{A.9}$$
$$g_4 = x_1\sqrt{x_3} - x_4 x_9\sqrt{\frac{40}{x_{11}}} > 0.$$

The objectives, decision variables, and constraints are defined as follows:

F_1: first product of combustion,

F_2: second product of combustion,

F_3: sum of third and fourth product of combustion,

x_1–x_{10}: number of moles of each product formed for each mole of propane burned,

x_{11}: sum of x_1 to x_{10},

g_1: first product of combustion,

g_2: second product of combustion,

g_3: third product of combustion,

g_4: fourth product of combustion.

Acknowledgments

This material is based on work supported by the National Science Foundation under Grant no. 0826124 and by a Dissertation Research Award from Cleveland State University. The comments of an anonymous reviewer were instrumental in improving this paper.

References

[1] P. Cilliers, *Complexity and Postmodernism: Understanding Complex Systems*, Routledge, New York, NY, USA, 1st edition, 1998.

[2] J. Allison, *Complex system optimization: a review of analytical target cascading, collaborative optimization, and other formulations [M.S. thesis]*, Mechanical Engineering Department, University of Michigan, Ann Arbor, Mich, USA, 2004.

[3] S. Bradley, A. Hax, and T. Magnanti, *Applied Mathematical Programming*, Addison Wesley, Reading, Mass, USA, 1977.

[4] W. Hammond, *Design Methodologies for Space Transportation Systems*, American Institution of Aeronautics & Astronautics, 2001.

[5] J. Martins and A. Lambe, "Multidisciplinary design optimization: a survey of architectures," *The AIAA Journal*, vol. 51, no. 9, pp. 2049–2075, 2013.

[6] E. J. Cramer, J. E. Dennis,, P. D. Frank, R. M. Lewis, and G. R. Shubin, "Problem formulation for multidisciplinary optimization," *SIAM Journal on Optimization*, vol. 4, no. 4, pp. 754–776, 1994.

[7] S. Kodiyalam and J. Sobieszczanski-Sobieski, "Multidisciplinary design optimization—some formal methods, framework requirements, and application to vehicle design," *The International Journal of Vehicle Design*, vol. 25, no. 1-2, pp. 3–22, 2001.

[8] R. Braun, P. Gage, and I. Kroo, "Implementation and performance issues in collaborative optimization," in *Proceedings of the AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Bellevue, Washington, DC, USA, 1996.

[9] T. Zang and L. Green, "Multidisciplinary design optimization techniques: implications and opportunities for fluid dynamics research," in *Proceedings of the 30th AIAA Fluid Dynamics Conference*, Norfolk, Va, USA, 1999.

[10] D. Simon, "Biogeography-based optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 6, pp. 702–713, 2008.

[11] D. Du, D. Simon, and M. Ergezer, "Biogeography-based optimization combined with evolutionary strategy and immigration refusal," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pp. 997–1002, San Antonio, Tex, USA, 2009.

[12] D. Simon, *Evolutionary Optimization Algorithms*, John Wiley & Sons, New York, NY, USA, 2013.

[13] R. Tanese, "Distributed genetic algorithms," in *Proceedings of the 3rd International Conference on Genetic Algorithms*, pp. 434–439, Fairfax, Va, USA, 1989.

[14] D. Whitley, "An Executable model of a simple genetic algorithm," in *Foundations of Genetic Algorithms 2*, D. Whitley, Ed., pp. 45–62, Morgan Kaufmann, Boston, Mass, USA, 1993.

[15] R. Schaefer, A. Byrski, and M. Smołka, "The island model as a Markov dynamic system," *The International Journal of Applied Mathematics and Computer Science*, vol. 22, no. 4, pp. 971–984, 2012.

[16] D. Whitley, S. Rana, and R. Heckendorn, "The island model genetic algorithm: on separability, population size and convergence," *Journal of Computing and Information Technology*, vol. 7, pp. 33–47, 1998.

[17] D. Whitley, "An overview of evolutionary algorithms: practical issues and common pitfalls," *Information and Software Technology*, vol. 43, no. 14, pp. 817–831, 2001.

[18] J. Laessig and D. Sudholt, "Design and analysis of migration in parallel evolutionary algorithms," *Soft Computing*, vol. 17, no. 7, pp. 1121–1144, 2013.

[19] N. Chakraborti and A. Kumar, "The optimal scheduling of a reversing strip mill: studies using multipopulation genetic algorithms and differential evolution," *Materials and Manufacturing Processes*, vol. 18, no. 3, pp. 433–445, 2003.

[20] G. Luque and E. Alba, *Parallel Genetic Algorithms: Theory and Real World Applications*, Springer, Berlin, Germany, 2011.

[21] O. Chikumbo and I. Nicholas, “Efficient thinning regimes for *Eucalyptus fastigata*: multi-objective stand-level optimisation using the island model genetic algorithm,” *Ecological Modelling*, vol. 222, no. 10, pp. 1683–1695, 2011.

[22] R. Mukherjee and S. Chakraborty, “Selection of EDM process parameters using biogeography-based optimization algorithm,” *Materials and Manufacturing Processes*, vol. 27, no. 9, pp. 954–962, 2012.

[23] R. Rarick, D. Simon, F. E. Villaseca, and B. Vyakaranam, “Biogeography-based optimization and the solution of the power flow problem,” in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pp. 1003–1008, San Antonio, Tex, USA, 2009.

[24] G. Thomas, P. Lozovyy, and D. Simon, “Fuzzy robot controller tuning with biogeography-based optimization,” in *Proceedings of the 24th International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems*, pp. 319–327, Syracuse, NY, USA, 2011.

[25] S. Nikumbh, S. Ghosh, and V. Jayaraman, “Biogeography-based informative gene selection and cancer classification using SVM and random forests,” in *Proceedings of the IEEE World Congress on Computational Intelligence*, pp. 187–192, Brisbane, Australia, 2012.

[26] W. Gong, Z. Cai, and C. X. Ling, “DE/BBO: a hybrid differential evolution with biogeography-based optimization for global numerical optimization,” *Soft Computing*, vol. 15, no. 4, pp. 645–665, 2011.

[27] L. Goel, D. Gupta, and V. K. Panchal, “Hybrid bio-inspired techniques for land cover feature extraction: a remote sensing perspective,” *Applied Soft Computing Journal*, vol. 12, no. 2, pp. 832–849, 2012.

[28] H. Kundra and M. Sood, “Cross-country path finding using Hybrid approach of PSO and BBO,” *International Journal of Computer Applications*, vol. 7, pp. 15–19, 2010.

[29] P. Arora, H. Kundra, and V. Panchal, “Fusion of biogeography based optimization and artificial bee colony for identification of natural terrain features,” *International Journal of Advanced Computer Science and Applications*, vol. 3, pp. 107–111, 2012.

[30] G. Wang, L. Guo, H. Duan, H. Wang, L. Liu, and M. Shao, “Hybridizing harmony search with biogeography based optimization for global numerical optimization,” *Journal of Computational and Theoretical Nanoscience*, vol. 10, pp. 2312–2322, 2013.

[31] H. Ma, M. Fei, D. Simon, and M. Yu, “Biogeography-based optimization for noisy fitness functions,” submitted for publication, 2013, http://academic.csuohio.edu/simond/bbo/noisy.

[32] P. K. Roy, S. P. Ghoshal, and S. S. Thakur, “Biogeography based optimization for multi-constraint optimal power flow with emission and non-smooth cost function,” *Expert Systems with Applications*, vol. 37, no. 12, pp. 8221–8228, 2010.

[33] Y. Song, M. Liu, and Z. Wang, “Biogeography-based optimization for the traveling salesman problems,” in *Proceedings of the 3rd International Joint Conference on Computational Sciences and Optimization*, pp. 295–299, Huangshan, China, 2010.

[34] P. K. Roy, S. P. Ghoshal, and S. S. Thakur, “Multi-objective optimal power flow using biogeography-based optimization,” *Electric Power Components and Systems*, vol. 38, no. 12, pp. 1406–1426, 2010.

[35] A. Bhattacharya and P. K. Chattopadhyay, “Application of biogeography-based optimization for solving multi-objective economic emission load dispatch problems,” *Electric Power Components and Systems*, vol. 38, no. 3, pp. 340–365, 2010.

[36] P. K. Roy, S. P. Ghoshal, and S. S. Thakur, “Biogeography based optimization technique applied to multi-constmints economic load dispatch problems,” in *Proceedings of the Transmission and Distribution Conference and Exposition: Asia and Pacific*, Seoul, Repuplic of Korea, October 2009.

[37] J. Abell and D. Du, “A framework for multiobjective, biogeography-based optimization of complex system families,” in *Proceedings of the AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*, Fort Worth, Tex, USA, 2010.

[38] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[39] D. Goldberg, *Genetic Algorithm in Search, Optimization, and Machine Learning*, Addison-Wesley, Boston, Mass, USA, 1989.

[40] N. Srinivas and K. Deb, “Multiobjective optimization using nondominated sorting in genetic algorithms,” *IEEE Transactions on Evolutionary Computation*, vol. 2, no. 3, pp. 221–248, 1994.

[41] M. Fonseca and P. Fleming, “Genetic algorithms for multiobjective optimization: formulation, discussion and generalization,” in *Proceedings of the International Conference on Genetic Algorithms*, pp. 416–423, Urbana-Champaign, Ill, USA, 1993.

[42] R. J. Hathaway and J. C. Bezdek, “Fuzzy c-means clustering of incomplete data,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 31, no. 5, pp. 735–744, 2001.

[43] P. Austin, *Cracking the Roulette Wheel: The System & Story of the CPA Who Cracked the Roulette Wheel*, CreateSpace Independent Publishing Platform, 3rd edition, 2010.

[44] S. Kodiyalam, “Evaluation of methods for multidisciplinary design optimization (MDO), phase I,” NASA CR-1998-208716, National Aeronautics and Space Administration, Langley Research Center, Hampton, Va, USA, 1998.

[45] X. Chen, B. Li, and Y. Lin, “Multidisciplinary design optimization with a new effective method,” *The Chinese Journal of Mechanical Engineering*, vol. 23, no. 4, pp. 505–510, 2010.

[46] M. Xiao, L. Gao, H. B. Qiu, X. Y. Shao, and X. Z. Chu, “An approach based on enhanced collaborative optimization and kriging approximation in multidisciplinary design optimization,” *Advanced Materials Research*, vol. 118–120, pp. 399–403, 2010.

[47] K. Jamuna and K. S. Swarup, “Multi-objective biogeography based optimization for optimal PMU placement,” *Applied Soft Computing Journal*, vol. 12, no. 5, pp. 1503–1510, 2012.

[48] P. K. Roy and D. Mandal, “Quasi-oppositional biogeography-based optimization for multi-objective optimal power flow,” *Electric Power Components and Systems*, vol. 40, no. 2, pp. 236–256, 2011.

[49] G. Rudolph, “Convergence analysis of canonical genetic algorithms,” *IEEE Transactions on Neural Networks*, vol. 5, no. 1, pp. 96–101, 1994.

[50] D. Simon, M. Ergezer, D. Du, and R. Rarick, “Markov models for biogeography-based optimization,” *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 41, no. 1, pp. 299–306, 2011.

[51] L. Padula, N. Alexandrov, and L. Green, “MDO test suite at NASA Langley research center,” in *Proceedings of the AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Bellevue, Washington, DC, USA, 1996.

[52] A. P. Morgan, A. Sommese, and L. T. Watson, "Mathematical reduction of a heart dipole model," *Journal of Computational and Applied Mathematics*, vol. 27, no. 3, pp. 407–410, 1989.

[53] N. P. Tedford and J. R. R. A. Martins, "Benchmarking multidisciplinary design optimization algorithms," *Optimization and Engineering*, vol. 11, no. 1, pp. 159–183, 2010.