Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 672983, 9 pages
http://dx.doi.org/10.1155/2014/672983

Research Article

Hybrid Biogeography-Based Optimization for Integer Programming

Zhi-Cheng Wang and Xiao-Bei Wu

College of Electronics and Information Engineering, Tongji University, Shanghai 201804, China

Correspondence should be addressed to Xiao-Bei Wu; xb.wu@ymail.com

Received 15 January 2014; Accepted 17 March 2014; Published 3 June 2014

Academic Editors: S. Balochian and Y. Zhang

Biogeography-based optimization (BBO) is a relatively new bioinspired heuristic for global optimization based on the mathematical models of biogeography. By investigating the applicability and performance of BBO for integer programming, we find that the original BBO algorithm does not perform well on a set of benchmark integer programming problems. Thus we modify the mutation operator and/or the neighborhood structure of the algorithm, resulting in three new BBO-based methods, named BlendBBO, BBO_DE, and LBBO_LDE, respectively. Computational experiments show that these methods are competitive approaches to solve integer programming problems, and the LBBO_LDE shows the best performance on the benchmark problems.

1. Introduction

An integer programming problem is a discrete optimization problem where the decision variables are restricted to integer values. In computer science and operations research, a remarkably wide range of problems, such as project scheduling, capital budgeting, goods distribution, and machine scheduling, can be expressed as integer programming problems [1–6]. Integer programming also has applications in bioinspired computational models such as artificial neural networks [7, 8].

The general form of an integer programming model can be stated as

$$\begin{aligned} \min \quad & f(\vec{x}) \\ \text{s.t.} \quad & \vec{x} \in \vec{S} \subseteq \mathbb{Z}^D, \end{aligned} \tag{1}$$

where $\vec{x}$ is a D-dimensional integer vector, $\mathbb{Z}^D$ is a D-dimensional discrete space of integers, and $\vec{S}$ is a feasible region that is not necessarily bounded. Any maximization version of integer programming problems can be easily transformed to a minimization problem.

One of the most well-known deterministic approaches for solving integer programming problems is the branch-and-bound algorithm [9]. It uses a "divide-and-conquer" strategy to split the feasible region into subregions, obtaining for each subregion a lower bound by ignoring the integrality constraints and checking whether the corresponding solution is a feasible one; if so, the current solution is optimum to the original problem; otherwise recursively split and tackle the subregions until all the variables are fixed to integers.

However, integer programming is known to be *NP*-hard [10], and thus the computational cost of deterministic algorithms increases very rapidly with problem size. In recent years, evolutionary algorithms (EA), which are stochastic search methods inspired by the principles of natural biological evolution, have attracted great attention and have been successfully applied to a wide range of computationally difficult problems. These heuristic algorithms do not guarantee finding the exact optimal solution in a single simulation run, but in most cases they are capable of finding acceptable solutions in a reasonable computational time.

Genetic algorithms (GA) are one of the most popular EA, but the encoding of the integer search space with fixed length binary strings as used in standard GA is not feasible for integer problems [11]. Many other heuristics, such as evolutionary strategy (ES) [12], particle swarm optimization (PSO) [13], and differential evolution (DE) [14], are initially proposed for continuous optimization problems. However, they can be adapted to integer programming by embedding the integer space into the real space and truncating or rounding real values to integers, and the applicability and performance of such approach are demonstrated by experimental studies.

Kelahan and Gaddy [15] conducted an early study that performs random search in integer spaces in the spirit of a (1 + 1)-ES; that is, at each iteration a child solution vector is generated by adding a random vector to the parent vector, and the better one between the parent and the child is kept for the next generation. Rudolph [11] developed a $(\mu + \lambda)$-ES based algorithm, which uses the principle of maximum entropy to guide the construction of a mutation distribution for arbitrary search spaces.

Laskari et al. [16] studied the ability of PSO for solving integer programming problems. On their test problems, PSO outperforms the branch-and-bound method in terms of number of function evaluations (NFE), and PSO exhibits high success rates even in cases where the branch-and-bound algorithm fails. Improved versions of PSO, including the quantum-behaved PSO which is based on the principle of state superposition and uncertainty [17] and barebones PSO which is based on samples from a normal distribution and requires no parameter tuning [18], have also been applied and shown to be efficient alternatives to integer programming problems.

Omran and Engelbrecht [19] investigated the performance of DE in integer programming. They tested three versions of DE and found that the self-adaptive DE (SDE) requiring no parameter tuning is the most efficient and performs better than PSO.

In this paper, we propose three algorithms for integer programming based on a relatively new bioinspired method, namely, biogeography-based optimization (BBO). We modify the mutation operator of the original BBO to enhance its exploration or global search ability and adopt a local neighborhood structure to avoid premature convergence. Experimental results show that our methods are competitive approaches to solving integer programming problems.

2. Biogeography-Based Optimization

Biogeography is the science of the geographical distribution of biological organisms over space and time. MacArthur and Wilson [20] established the mathematical models of island biogeography, which show that the species richness of an island can be predicted in terms of such factors as habitat area, immigration rate, and extinction rate. Inspired by this, Simon [21] developed the BBO algorithm, where a solution vector is analogous to a habitat, the solution components are analogous to a set of suitability index variables (SIVs), and the solution fitness is analogous to the species richness or habitat suitability index (HSI) of the habitat. Central to the algorithm is the equilibrium theory of island biogeography, which indicates that high HSI habitats have a high species emigration rate and low HSI habitats have a high species immigration rate. For example, in a linear model of species richness (as illustrated in Figure 1), a habitat H_i's immigration rate λ_i and emigration rate μ_i are calculated based on its fitness f_i as follows:

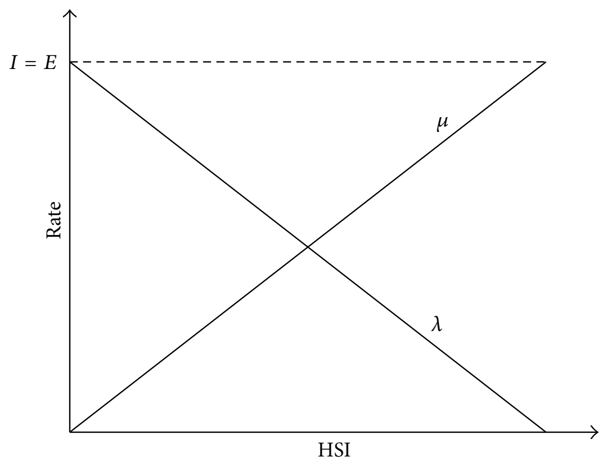

FIGURE 1: A linear model of emigration and immigration rates of a habitat.

$$\lambda_i = I\left(\frac{f_{\max} - f_i}{f_{\max} - f_{\min}}\right)$$
$$\mu_i = E\left(\frac{f_i - f_{\min}}{f_{\max} - f_{\min}}\right), \tag{2}$$

where $f_{\max}$ and $f_{\min}$ are, respectively, the maximum and minimum fitness values among the population and I and E are, respectively, the maximum possible immigration rate and emigration rate. However, there are other nonlinear mathematical models of biogeography that can be used for calculating the migration rates [22, 23].

Migration is used to modify habitats by mixing features within the population. BBO also has a mutation operator for changing SIV within a habitat itself and thus probably increasing diversity of the population. For each habitat H_i, a species count probability P_i computed from λ_i and μ_i indicates the likelihood that the habitat was expected a priori to exist as a solution for the problem. In this context, very high HSI habitats and very low HSI habitats are both equally improbable, and medium HSI habitats are relatively probable. The mutation rate of habitat H_i is inversely proportional to its probability:

$$\pi_i = \pi_{\max}\left(1 - \frac{P_i}{P_{\max}}\right), \tag{3}$$

where $\pi_{\max}$ is a control parameter and $P_{\max}$ is the maximum habitat probability in the population.

Algorithm 1 describes the general framework of BBO for a D-dimensional global numerical optimization problem (where l_d and u_d are the lower and upper bounds of the dth dimension, respectively, and rand is a function that generates a random value uniformly distributed in $[0, 1]$).

```
(1) Randomly initialize a population of $n$ solutions (habitats);
(2) while stop criterion is not satisfied do
(3)     for $i = 1$ to $n$ do
(4)         Calculate $\lambda_i$, $\mu_i$, and $\pi_i$ according to $f_i$;
(5)     for $i = 1$ to $n$ do
(6)         for $d = 1$ to $D$ do
(7)             if rand() < $\lambda_i$ then //migration
(8)                 Select a habitat $H_j$ with probability $\propto \mu_j$;
(9)                 $H_{i,d} \leftarrow H_{j,d}$;
(10)    for $i = 1$ to $n$ do
(11)        for $d = 1$ to $D$ do
(12)            if rand() < $\pi_i$ then //mutation
(13)                $H_{i,d} \leftarrow l_d + \text{rand}() \times (u_d - l_d)$;
(14)    Evaluate the fitness values of the habitats;
(15)    Update $f_{\max}$, $P_{\max}$, and the best known solution;
(16) return the best known solution.
```

Algorithm 1: The original BBO algorithm.

Typically, in Line 8 we can use a roulette wheel method for selection, the time complexity of which is $O(n)$. It is not difficult to see that the complexity of each iteration of the algorithm is $O(n^2D + nO(f))$, where $O(f)$ is the time complexity for computing the fitness function f.

3. Biogeography-Based Heuristics for Integer Programming

In BBO, the migration operator provides good exploitation ability, while the broader exploration of the search space is mainly based on the mutation operator. Simon [21] suggested that $\pi_{\max}$ should be set to a small value (about 0.01), which results in low mutation rates. However, when being applied to integer programming, we need to use higher mutation rates to improve the exploration of search space. According to our experimental studies, when $\pi_{\max}$ is set to about 0.25~0.3, the BBO algorithm exhibits the best performance on integer programming problems.

Note that the migration operator does not violate the integer constraints, and the rounding of real values to integers is required only after mutations (Line 13 of Algorithm 1). Nevertheless, even using a higher mutation rate, the performance of BBO is far from satisfactory for integer programming. This is mainly because random mutation operator does not utilize any information of the population to guide the exploration of search space. In this work, we introduce two other mutation operators to BBO, which results in three variants of BBO for integer programming.

3.1. A Blended Mutation Operator. In the first variant, namely, BlendBBO, we use a blended mutation operator, which is motivated by the blended crossover operator used by Mühlenbein and Schlierkamp-Voosen [24] in GA and by Ma and Simon [25] in constrained optimization. In our approach, if a component of vector H_i is subject to mutate, we first select another vector H_j with probability $\propto \mu_j$ and then use the following equation to work out the new value of the component:

$$H_{i,d} = \text{round}\left(\alpha H_{i,d} + (1-\alpha) H_{j,d}\right), \quad (4)$$

where α is a random value uniformly distributed in $[0,1]$. Note that if the dth dimension of the search space has a bound, (4) will never result in a value outside the bound.

Moreover, we employ an elitism mechanism in solution update (as used in ES [12, 26]): the migration operator always generates a new vector H_i' for each existing vector H_i (rather than directly changing H_i); if H_i' is better than H_i, no mutation will be applied and H_i' directly enters to the next generation; otherwise the mutation operator is applied to H_i. This not only decreases the required NFE but also increases the convergence speed of the algorithm. The algorithm flow of BBO with the blended mutation is presented in Algorithm 2.

3.2. DE Mutation Operator. The second variant, namely, BBO_DE, replaces the random mutation operator with the mutation operator of DE, which mutates a vector component by adding the weighted difference between the corresponding components of two randomly selected vectors to a third one:

$$H_{i,d} = \text{round}\left(H_{r_1,d} + F\left(H_{r_2,d} - H_{r_3,d}\right)\right), \quad (5)$$

where r_1, r_2, and r_3 are three unique randomly selected habitat indices that are different to i, and F is a constant scaling coefficient.

DE is well known for its good exploration ability, and the combination of BBO migration and DE mutation achieves a good balance between exploitation and exploration. BBO_DE also uses our new solution update mechanism described above. Therefore, the algorithm flow of BBO_DE simply replaces Lines 15 and 16 of Algorithm 2 with the DE mutation operation described by (5).

```
(1) Randomly initialize a population of $n$ solutions (habitats);
(2) while stop criterion is not satisfied do
(3)     for $i = 1$ to $n$ do
(4)         Calculate $\lambda_i$, $\mu_i$, and $\pi_i$ according to $f_i$;
(5)     for $i = 1$ to $n$ do
(6)         Let $H_i' = H_i$;
(7)         for $d = 1$ to $D$ do
(8)             if rand() < $\lambda_i$ then //migration
(9)                 Select a habitat $H_j$ with probability $\propto \mu_j$;
(10)                $H_{i,d}' \leftarrow H_{j,d}$;
(11)        if $f(H_i) < f(H_i')$ then $H_i \leftarrow H_i'$;
(12)        else
(13)            for $d = 1$ to $D$ do
(14)                if rand() < $\pi_i$ then //mutation
(15)                    Let $\alpha$ = rand() and select a habitat $H_j$ with probability $\propto \mu_j$;
(16)                    $H_{i,d} \leftarrow$ round($\alpha H_{i,d} + (1-\alpha)H_{j,d}$);
(17)    Evaluate the fitness values of the habitats;
(18)    Update $f_{\max}$, $P_{\max}$, and the best known solution;
(19) return the best known solution.
```

Algorithm 2: The BBO with the blended mutation for integer programming.

3.3. Local Topology of the Population. The original BBO uses a fully connected topology; that is, all the individual solutions are directly connected in the population and can migrate with each other. But such a global topology is computationally intensive and is prone to premature convergence. To overcome this problem, our third variant replaces the global topology with a local one. One of the simplest local topologies is the ring topology, where each individual is directly connected to two other individuals [27, 28]. But here we employ a more generalized local topology, the random topology, where each individual has K immediate neighbors that are randomly selected from the population and K is a control parameter [28].

In consequence, whenever an individual vector H_i is to be immigrated, the emigrating vector is chosen from its neighbors rather than the whole population, based on the migration rates. The neighborhood structure can be saved in an $n \times n$ matrix L: if two habitats H_i and H_j are directly connected then $L(i, j) = 1$; otherwise $L(i, j) = 0$. It is easy to see that the complexity of each iteration of the algorithm is $O(nKD + nO(f))$.

Storn and Price [14] have proposed several different strategies on DE mutation. The scheme of (5) is denoted as DE/rand/1. Another scheme is named DE/best/1, which always chooses the best individual of the population as H_{r_1} in (5). Omran et al. [29] extended it to the DE/lbest/1 scheme, which uses a ring topology and always chooses the better neighbor of the vector to be mutated.

In our approach, BBO migration and DE mutation share the same local random topology. That is, each H_i individual has K neighbors, and at each time an H_j is chosen from the neighbors with probability $\propto \mu_j$ to participate in the mutation such that

$$H_{i,d} = \text{round}\left(H_{j,d} + F\left(H_{r_2,d} - H_{r_3,d}\right)\right). \tag{6}$$

Moreover, if the current best solution has not been improved after every n_p generation (where n_p is a predefined constant), we reset the neighborhood structure randomly.

The third variant is named LBBO_LDE, and it also uses the same solution update mechanism as the previous two variants.

4. Computational Experiments

We test the three variants of BBO on a set of integer programming benchmark problems, which are taken from [16, 30, 31] and frequently encountered in the relevant literature. The details of the benchmark problems are described in the Appendix. For comparison, we also implement the basic BBO, DE, and SDE [19] for integer programming. The branch-and-bound method is not included for comparison, because it has shown that DE outperforms branch-and-bound on most test problems [16, 19].

For all the six algorithms, we use the same population size $n = 50$ and run them on each problem for 40 times with different random seeds. The migration control parameters are set as $I = E = 1$ for BBO, BlendBBO, BBO_DE, and LBBO_LDE, and the mutation control parameter $\pi_{\max}$ is set to 0.01 for BBO and 0.25 for BlendBBO (BBO_DE and LBBO_LDE do not use this parameter). Empirically, the neighborhood size K and the threshold of nonimprovement generations n_p are both set to 3 for LBBO_LDE. The other parameters with regard to DE and SDE are set as suggested in [14, 32].

The first two problems F_1 and F_2 are high-dimensional problems. For F_1, we, respectively, consider it in 10 and 30 dimensions. Table 1 presents the success rates (SR) and required NFE of the algorithms to achieve the optimum in 10 dimensions, and Figure 2 presents the corresponding

Table 1: SR and required NFE of the algorithms on F_1 in 10 dimensions.

Method	SR	Best	Worst	Mean	Std
BBO	0%	NA	NA	NA	NA
BlendBBO	20%	1946	2215	2080.50	190.21
DE	100%	3200	4100	3777.50	239.78
SDE	100%	3050	4050	3762.40	265.23
BBO_DE	100%	3102	3964	3674.80	269.07
LBBO_LDE	100%	2061	2694	2493.75	145.59

Table 2: SR and required NFE of the algorithms on F_1 in 30 dimensions.

Method	SR	Best	Worst	Mean	Std
BBO	0%	NA	NA	NA	NA
BlendBBO	0%	NA	NA	NA	NA
DE	0%	NA	NA	NA	NA
SDE	85%	7850	11350	9301.00	1130.10
BBO_DE	90%	6765	9752	8204.25	920.39
LBBO_LDE	100%	5923	7071	6471.60	272.29

convergence curves of the algorithms. As we can see, the original BBO fails to solve the problem, and the SR of BlendBBO is only 20%. The four algorithms utilizing the DE mutation operator can guarantee the optimal result on the 10-dimensional problem, among which LBBO_LDE shows the best performance, and the other three algorithms have similar performance, but the result of BBO_DE is slightly better than DE and SDE.

Table 2 and Figure 3, respectively, present the results and the convergence curves of the algorithms on F_1 in 30 dimensions. On this high-dimensional problem, BBO, BlendBBO, and DE all fail to obtain the optimum, SDE and BBO_DE, respectively, have SR of 85% and 90% for obtaining the optimum, and only our LBBO_LDE can always guarantee the optimum.

From the convergence curves we can also find that the BBO algorithm converges very fast at the early stage, but thereafter its performance deteriorates because it is ineffective to explore other potentially promising areas of the search space. By combining with the DE mutation operator, our hybrid BBO methods inherit the fast convergence speed of BBO, at the same time taking advantage of the exploration ability of DE.

For F_2, we, respectively, consider it in 5 and 15 dimensions, the experimental results of which are, respectively, presented in Tables 3 and 4 and the convergence curves of which are presented in Figures 4 and 5. The results are similar to those of F_1: for the low dimensional problem, SDE, BBO_DE, and LBBO_LDE are efficient; for the high-dimensional problem, only LBBO_LDE can guarantee the optimum; the performance LBBO_LDE is the best while that of BBO is the worst; SDE performs better than DE and BBO_DE performs slightly better than SDE, and BlendBBO outperforms BBO but is worse than the algorithms with the DE mutation operator.

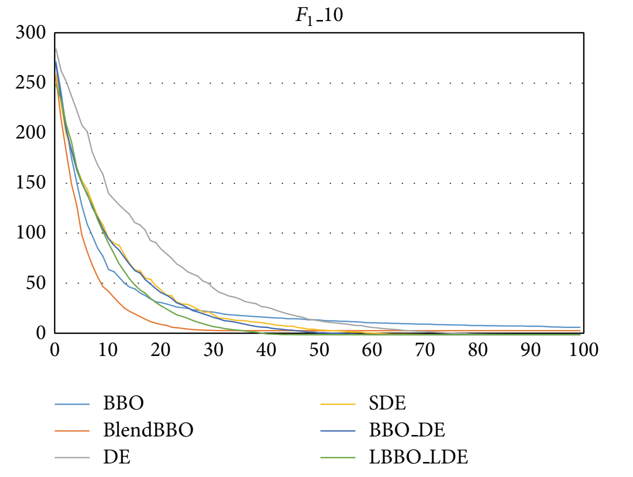

Figure 2: The convergence curves of the algorithms on F_1 in 10 dimensions, where the vertical axis represents the objective value and the horizontal axis represents the number of generations.

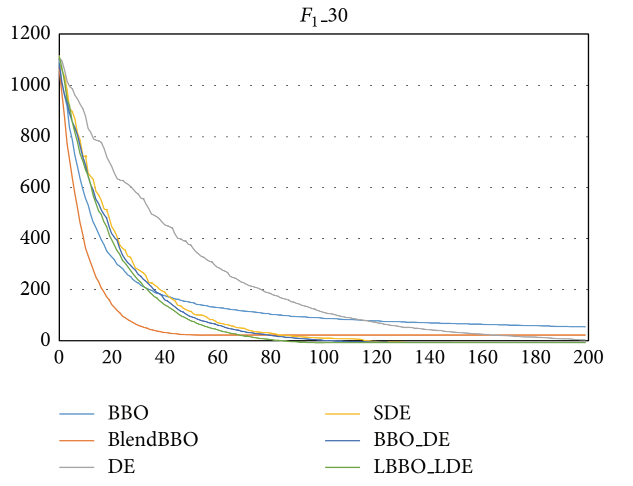

Figure 3: The convergence curves of the algorithms on F_1 in 30 dimensions, where the vertical axis represents the objective value and the horizontal axis represents the number of generations.

Table 3: SR and required NFE of the algorithms on F_2 in 5 dimensions.

Method	SR	Best	Worst	Mean	Std
BBO	10%	3663	5168	4415.50	1064.20
BlendBBO	55%	1154	1545	1363.36	121.94
DE	95%	1500	2100	1797.37	188.91
SDE	100%	1450	2600	2022.35	368.17
BBO_DE	100%	1773	2669	2336.65	258.22
LBBO_LDE	100%	1058	1898	1451.20	214.48

F_3 is a 5-dimensional problem more difficult than F_2. As we can see from the results shown in Table 5, BBO and BlendBBO always fail on the problem, and DE, SDE, and LBBO_LDE can guarantee the optimum. The required

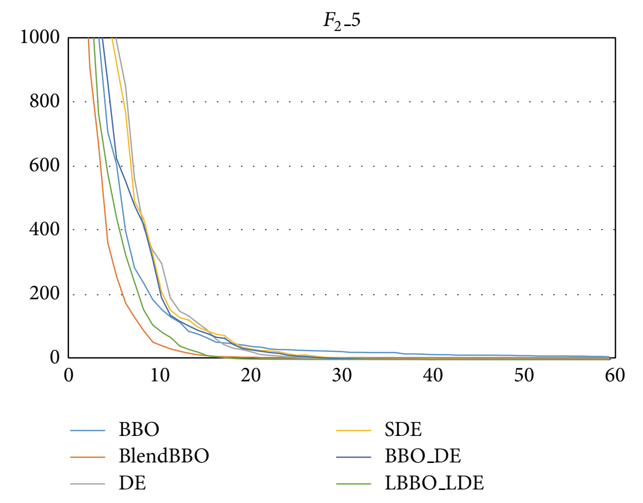

Figure 4: The convergence curves of the algorithms on F_2 in 5 dimensions, where the vertical axis represents the objective value and the horizontal axis represents the number of generations.

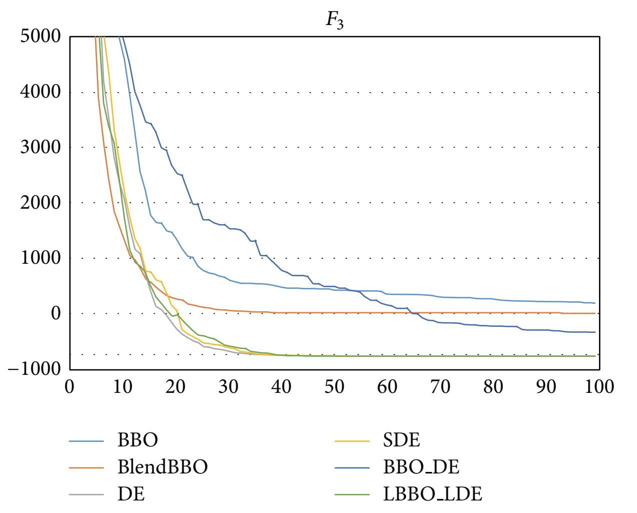

Figure 6: The convergence curves of the algorithms on F_3, where the vertical axis represents the objective value and the horizontal axis represents the number of generations.

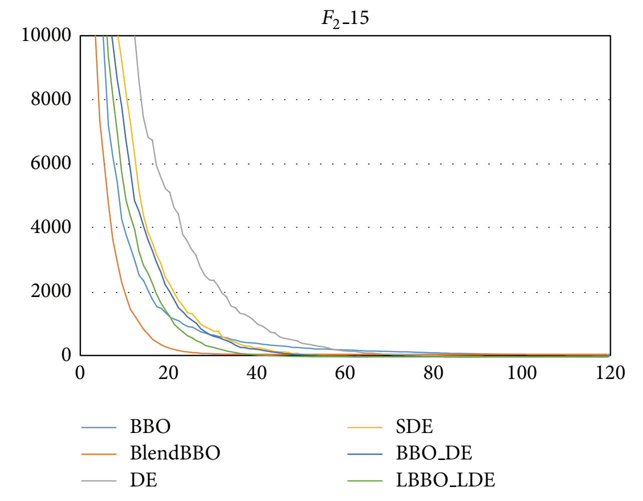

Figure 5: The convergence curves of the algorithms on F_2 in 15 dimensions, where the vertical axis represents the objective value and the horizontal axis represents the number of generations.

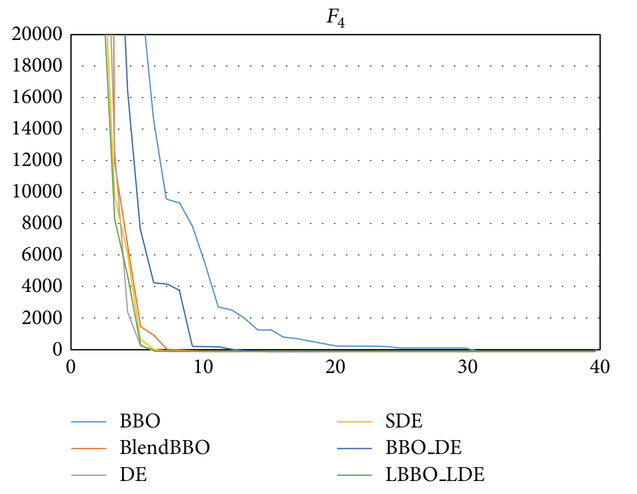

Figure 7: The convergence curves of the algorithms on F_4, where the vertical axis represents the objective value and the horizontal axis represents the number of generations.

Table 4: SR and required NFE of the algorithms on F_2 in 15 dimensions.

Method	SR	Best	Worst	Mean	Std
BBO	0%	NA	NA	NA	NA
BlendBBO	2.5%	4815	4815	4815	NA
DE	95%	5300	6650	5978.95	335.95
SDE	95%	5000	6400	5698.32	389.24
BBO_DE	97.5%	5030	6210	5639.11	362.69
LBBO_LDE	100%	3488	4528	4188.30	300.77

NFE of DE is slightly better than SDE and LBBO_LDE, but LBBO_LDE converges faster than DE, as shown in Figure 6.

F_4 is a relatively easy problem, on which even the worst BBO has an SR of 75%, and all the other algorithms can guarantee the optimum. LBBO_LDE is the best one in terms of both NFE and convergence speed, as shown in Table 6 and Figure 7.

The remaining three test problems are also relatively easy. The experimental results are presented in Tables 7, 8, and 9, and the convergence curves are shown in Figures 8, 9, and 10, respectively. As we can clearly see, the four algorithms with the DE mutation operator can always obtain the optima on these problems, and LBBO_LDE shows the best performance.

In summary, our LBBO_LDE outperforms the other algorithms on all of the test problems. Generally speaking, the original BBO converges fast at first, but it is easy to be trapped by the local optima. BlendBBO alleviates the dilemma to a certain degree, but the DE mutation operator is more effective than the blended mutation operator, as

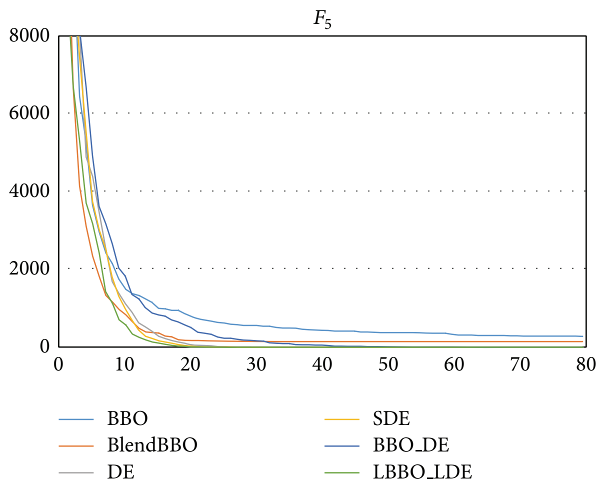

Figure 8: The convergence curves of the algorithms on F_5, where the vertical axis represents the objective value and the horizontal axis represents the number of generations.

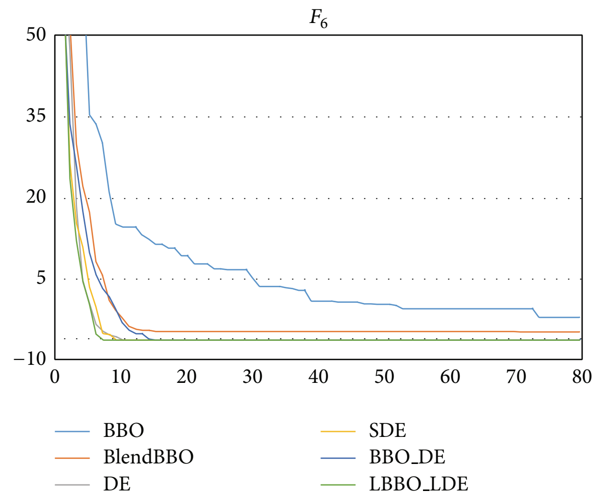

Figure 9: The convergence curves of the algorithms on F_6, where the vertical axis represents the objective value and the horizontal axis represents the number of generations.

Table 5: SR and required NFE of the algorithms on F_3.

Method	SR	Best	Worst	Mean	Std
BBO	0%	NA	NA	NA	NA
BlendBBO	0%	NA	NA	NA	NA
DE	100%	2050	2950	2490.00	262.38
SDE	100%	2500	4050	3260.00	638.05
BBO_DE	10%	4810	5246	5028.00	308.30
LBBO_LDE	100%	1758	4181	2958.85	849.24

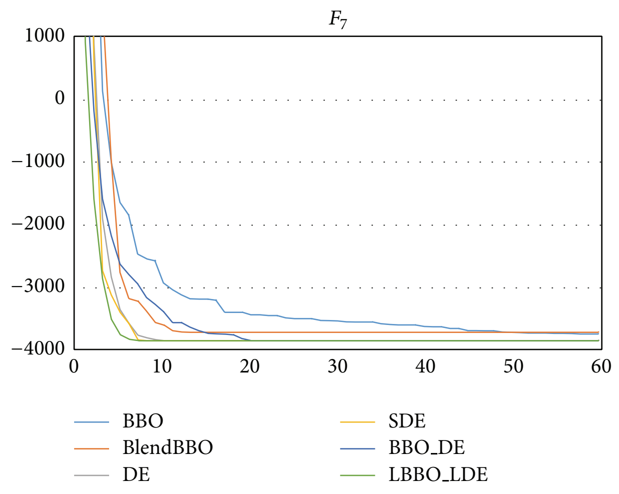

Figure 10: The convergence curves of the algorithms on F_7, where the vertical axis represents the objective value and the horizontal axis represents the number of generations.

Table 6: SR and required NFE of the algorithms on F_4.

Method	SR	Best	Worst	Mean	Std
BBO	75%	477	3429	1726.87	1020.19
BlendBBO	100%	258	552	424.50	85.34
DE	100%	300	650	420.00	93.75
SDE	100%	250	600	520.00	129.65
BBO_DE	10%	59	1058	632.30	277.77
LBBO_LDE	100%	236	525	400.60	72.20

Table 7: SR and required NFE of the algorithms on F_5.

Method	SR	Best	Worst	Mean	Std
BBO	0%	NA	NA	NA	NA
BlendBBO	25%	1223	2676	1842	692.53
DE	100%	1200	1700	1445.00	140.39
SDE	100%	1100	1750	1550.50	196.45
BBO_DE	100%	2473	4498	3681.25	620.36
LBBO_LDE	100%	1012	1874	1532.35	249.46

Table 8: SR and required NFE of the algorithms on F_6.

Method	SR	Best	Worst	Mean	Std
BBO	45%	140	1989	1150.22	678.16
BlendBBO	80%	178	635	455.81	128.81
DE	100%	200	550	392.50	140.39
SDE	100%	200	500	405.20	103.72
BBO_DE	100%	196	995	708.50	198.38
LBBO_LDE	100%	183	511	410.05	86.37

demonstrated by our experimental results. By combining BBO and DE, the BBO_DE algorithm provides an efficient alternative to popular methods such as SDE. The local topology used in LBBO_LDE further improves the search ability and suppresses the premature convergence, especially on high-dimensional problems where the performance of DE and SDE deteriorates quickly. Therefore, LBBO_LDE is a very competitive heuristic method for solving integer programming problem.

Table 9: SR and required NFE of the algorithms on F_7.

Method	SR	Best	Worst	Mean	Std
BBO	60%	167	3403	1714.75	986.18
BlendBBO	70%	262	721	459.50	149.00
DE	100%	350	600	480.00	76.78
SDE	100%	300	650	451.00	89.33
BBO_DE	100%	479	1327	978.40	311.11
LBBO_LDE	100%	249	614	389.45	96.08

5. Conclusion

In this paper we develop three algorithms for integer programming based on the BBO heuristic. The BlendBBO uses a blended mutation operator, BBO_DE integrates the DE mutation operator, and LBBO_LDE further uses a local neighborhood structure for selecting individuals for migration and mutation. Experimental results show that LBBO_LDE has the best performance on a set of benchmark integer programming problem.

In general, the LBBO_LDE algorithm with local neighborhood size K of 3~5 is efficient on the test problem, but none of the values can provide the best performance on all the problems. Currently we are studying a mechanism that dynamically adjusts the neighborhood size as well as other control parameters according to the search state [33]. Moreover, the test problems considered in the paper only have bounds for decision variables but do not include other constraints, and we are extending the proposed approach to solve more complex constrained optimization problems, including multiobjective ones [34–36]. We also believe that our approach can be adapted to effectively handle other kinds of combinatorial optimization problems, such as 0-1 integer programming and permutation-based optimization.

Appendix

Integer Programming Benchmark Problems

Consider

$$F_1(\vec{x}) = \sum_{i=1}^{D} x_i, \quad \vec{x}^* = (0)^D, \quad F_1(\vec{x}^*) = 0,$$

$$F_2(\vec{x}) = \sum_{i=1}^{D} x_i^2, \quad \vec{x}^* = (0)^D, \quad F_2(\vec{x}^*) = 0,$$

$$F_3(\vec{x}) = -(15, 27, 36, 18, 12)\,\vec{x}^{\top} + \vec{x}\begin{pmatrix} 35 & -20 & -10 & 32 & -10 \\ -20 & 40 & -6 & -31 & 32 \\ -10 & -6 & 11 & -6 & -10 \\ 32 & -31 & -6 & 38 & -20 \\ -10 & 32 & -10 & -20 & 32 \end{pmatrix}\vec{x}^{\top},$$

$$\vec{x}^* = (0, 11, 22, 16, 6) \quad \text{or } \vec{x}^* = (0, 12, 23, 17, 6),$$

$$F_3(\vec{x}^*) = -737,$$

$$F_4(\vec{x}) = \left(9x_1^2 + 2x_2^2 - 11\right)^2 + \left(3x_1 + 4x_2^2 - 7\right)^2,$$

$$\vec{x}^* = (1, 1), \quad F_4(\vec{x}^*) = 0,$$

$$F_5(\vec{x}) = \left(x_1 + 10x_2\right)^2 + 5\left(x_3 - x_4\right)^2 + \left(x_2 - 2x_3\right)^4 + 10\left(x_1 - x_4\right)^4,$$

$$\vec{x}^* = (0)^4, \quad F_5(\vec{x}^*) = 0,$$

$$F_6(\vec{x}) = 2x_1^2 + 3x_2^2 + 4x_1x_2 - 6x_1 - 3x_2,$$

$$\vec{x}^* = (2, -1), \quad F_6(\vec{x}^*) = -6,$$

$$F_7(\vec{x}) = -3803.84 - 138.08x_1 - 232.93x_2 + 123.08x_1^2 + 203.64x_2^2 + 182.25x_1x_2,$$

$$\vec{x}^* = (0, 1), \quad F_7(\vec{x}^*) = 3833.12. \tag{A.1}$$

In the above problems, the ranges of variables are all set as $\vec{x} \in [-100, 100]^D$.

Conflict of Interests

The authors declare that they have no conflict of interests regarding the publication of this paper.

References

[1] S. Sofianopoulou, "The process allocation problem: a survey of theapplication of graph-theoretic and integer programming approaches," *Journal of the Operational Research Society*, vol. 43, no. 5, pp. 407–413, 1992.

[2] J. M. Whitacre, "Recent trends indicate rapid growth of nature-inspired optimization in academia and industry," *Computing*, vol. 93, no. 2-4, pp. 121–133, 2011.

[3] Y. Zheng and J. Xue, "A problem reduction based approach to discrete optimization algorithm design," *Computing*, vol. 88, no. 1-2, pp. 31–54, 2010.

[4] Y. J. Zheng, S. Y. Chen, Y. Lin, and W. L. Wang, "Bio-inspired optimizationof sustainable energy systems: a review," *Mathematical Problemsin Engineering*, vol. 2013, Article ID 354523, 12 pages, 2013.

[5] Y. J. Zheng, H. F. Ling, H. H. Shi, H. S. Chen, and S. Y. Chen, "Emergencyrailway wagon scheduling by hybrid biogeography-based optimization," *Computers & Operations Research*, vol. 43, no. 3, pp. 1–8, 2014.

[6] Y. J. Zheng, H. F. Ling, and J. Y. Xue, "Ecogeography-based optimization: enhancing biogeography-based optimization with ecogeographic barriers and differentiations," *Computers & Operations Research*, vol. 50, pp. 115–127, 2014.

[7] V. Dua, "A mixed-integer programming approach for optimal configuration of artificial neural networks," *Chemical Engineering Research and Design*, vol. 88, no. 1, pp. 55–60, 2010.

[8] Y. Tan, J. Wang, and J. M. Zurada, "Nonlinear blind source separation using a radial basis function network," *IEEE Transactions on Neural Networks*, vol. 12, no. 1, pp. 124–134, 2001.

[9] G. Nemhauser and L. Wolsey, *Integer and Combinatorial Optimization*, John Wiley & Sons, 1988.

[10] M. R. Garey, D. S. Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling," *Mathematics of Operations Research*, vol. 1, no. 2, pp. 117–129, 1976.

[11] G. Rudolph, "An evolutionary algorithm for integer programming," in *Parallel Problem Solving from Nature*, Y. Davidor, H. P. Schwefel, and R. Männer, Eds., vol. 866 of *Lecture Notes in Computer Science*, pp. 139–148, Springer, Berlin, 1994.

[12] H. G. Beyer and H. P. Schwefel, "Evolution strategies—a comprehensive introduction," *Natural Computing*, vol. 1, no. 1, pp. 3–52, 2002.

[13] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, December 1995.

[14] R. Storn and K. Price, "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[15] R. Kelahan and J. Gaddy, "Application of the adaptive random searchto discrete and mixed integer optimization," *International Journal for Numerical Methods in Engineering*, vol. 12, pp. 289–298, 1978.

[16] E. C. Laskari, K. E. Parsopoulos, and M. N. Vrahatis, "Particle swarm optimizationfor integer programming," in *Proceedings of the IEEE Congress on Evolutionary Computation*, vol. 2, pp. 1582–1587, 2002.

[17] J. Liu, J. Sun, and W. Xu, "Quantum-behaved particle swarm optimizationfor integer programming," in *Neural Information Processing*, I. King, J. Wang, L. W. Chan, and D. Wang, Eds., vol. 4233 of *Lecture Notes in Computer Science*, pp. 1042–1050, Springer, Berlin, Germany, 2006.

[18] M. G. H. Omran, A. Engelbrecht, and A. Salman, "Barebones particle swarm for integer programming problems," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '07)*, pp. 170–175, April 2007.

[19] M. G. H. Omran and A. P. Engelbrecht, "Differential evolution for integer programming problems," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '07)*, pp. 2237–2242, September 2007.

[20] R. MacArthur and E. Wilson, *The Theory of Biogeography*, Princeton University Press, Princeton, NJ, USA, 1967.

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

[22] H. Ma, "An analysis of the equilibrium of migration models for biogeography-based optimization," *Information Sciences*, vol. 180, no. 18, pp. 3444–3464, 2010.

[23] H. Ma and D. Simon, "Analysis of migration models of biogeography-based optimization using Markov theory," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 6, pp. 1052–1060, 2011.

[24] H. Mühlenbein and D. Schlierkamp-Voosen, "Predictive models for thebreeder genetic algorithm I. Continuous parameter optimization," *Evolutionary Computation*, vol. 1, no. 1, pp. 25–49, 1993.

[25] H. Ma and D. Simon, "Blended biogeography-based optimization for constrained optimization," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 3, pp. 517–525, 2011.

[26] 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 (SMC '09)*, pp. 997–1002, October 2009.

[27] X. Li, "Niching without niching parameters: particle swarm optimization using a ring topology," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 1, pp. 150–169, 2010.

[28] Y. J. Zheng, H. F. Ling, X. B. Wu, and J. Y. Xue, "Localized biogeographybasedoptimization," *Soft Computing*, 2014.

[29] M. G. Omran, A. P. Engelbrecht, and A. Salman, "Using neighborhood topologies with differential evolution," in *Proceedings of the Workshop of International Conference on Computational Intelligence and Security*, pp. 35–40, 2005.

[30] A. Glankwahmdee, J. S. Liebman, and G. L. Hogg, "Unconstrained discrete nonlinear programming," *Engineering Optimization*, vol. 4, no. 2, pp. 95–107, 1979.

[31] S. Rao, *Engineering Optimization—Theory and Practice*, Wiley Eastern, New Delhi, India, 1996.

[32] M. Omran, A. Salman, and A. Engelbrecht, "Self-adaptive differential evolution," in *Computational Intelligence and Security*, Y. Hao, J. Liu, Y. Wang et al., Eds., vol. 3801 of *Lecture Notes in Computer Science*, pp. 192–199, Springer, Berlin, Germany, 2005.

[33] Y. J. Zheng, H. F. Ling, and Q. Guan, "Adaptive parameters for a modifiedcomprehensive learning particle swarm optimizer," *Mathematical Problemsin Engineering*, vol. 2012, Article ID 207318, 11 pages, 2012.

[34] Y. J. Zheng, Q. Song, and S. Y. Chen, "Multiobjective fireworks optimizationfor variable-rate fertilization in oil crop production," *Applied Soft Computing*, vol. 13, no. 11, pp. 4253–4263, 2013.

[35] Y. J. Zheng and S. Y. Chen, "Cooperative particle swarm optimization formultiobjective transportation planning," *Applied Intelligence*, vol. 39, no. 1, pp. 202–216, 2013.

[36] Y. Zheng, H. Ling, J. Xue, and S. Chen, "Population classificationin fire evacuation: a multiobjective particle swarm optimization approach," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 1, pp. 70–81, 2014.