Available online at www.tjnsa.com
J. Nonlinear Sci. Appl. 9 (2016), 1261–1284
Research Article

Journal of Nonlinear Science and Applications
Print: ISSN 2008-1898 Online: ISSN 2008-1901

Difference-genetic co-evolutionary algorithm for nonlinear mixed integer programming problems

Yuelin Gao[a,b], Ying Sun[b,*], Jun Wu[a]

[a] Institute of Information and System Science, Beifang University of Nationalities, Yinchuan, 750021, China.
[b] School of Computer Science and Information Engineering, Hefei University of Technology, Hefei, 230009, China.

Communicated by I. Argyros

Abstract

In this paper, the difference genetic co-evolutionary algorithm (D-GCE) is proposed for the mixed integer programming problems. First, the mixed integer programming problem with constrains converted to unconstrained bi-objective optimization problems. Secondly, selection mechanism combines the Pareto dominance and superiority of feasible solution methods to choose the excellent individual as the next generation. Final, differential evolution algorithm and genetic algorithm handle the continuous part and discrete part, respectively. Numerical experiments on 24 test functions have shown that the new approach is efficient. The comparison results among the D-GCE and other evolutionary algorithms indicate that the proposed D-GCE algorithm is competitive with and in some cases superior to, other existing algorithms in terms of the quality, efficiency, convergence rate, and robustness of the final solution. ©2016 All rights reserved.

Keywords: Mixed integer programming, differential evolution, genetic algorithm, co-evolution, constrained optimization.
2010 MSC: 90C11, 65J08, 65K10.

1. Introduction

Mathematical programming problem is an important branch in the field of operations research. With the development of operational research, scholars have summed up a lot of programming models. Some models is called integer nonlinear programming problems, in which some or all decision variables are restricted to have integer value or discrete value with nonlinear constraints or nonlinear objective. When mixed some

*Corresponding author
Email addresses: gaoyuelin@263.net (Yuelin Gao), nxsunying@126.com (Ying Sun), wujunmath@163.com (Jun Wu)

Received 2015-09-01

continuous variables in the models, this problem is mixed integer nonlinear programming problem. It has been widely applied to real-world such as power distribution system network optimization, engineering design problems, the conflict problems in air traffic management system, production scheduling, vehicle path planning and portfolio selection problem [10, 18, 20, 25, 31] and so on.

Mixed integer nonlinear programming problems can be expressed as follows:

$$\begin{cases} min & f(x,y), \\ s.t. & g_i(x,y) \leq 0, i = 1,2,...,m, \\ & h_j(x,y) = 0, j = 1,2,...,n, \\ & x^L \leq x \leq x^U, \\ & y^L \leq y \leq y^U, \\ & x = (x_1, x_2, ..., x_{nC}), \\ & y = (y_1, y_2, ..., y_{nI}). \end{cases} \tag{1.1}$$

Where x is a nC-dimensional real variables and y is a nI-dimensional integer variables, x^L , x^U are the lower and upper limits of the real variables, y^L , y^U are the lower and upper limits of discrete variables, $f(x,y)$ is the objective function to be optimized, $g_i(x,y)$ is the set of inequality constraints, $h_j(x,y)$ is the set of equality constraints. The m and n are the number of inequality constraints and equality constraints, respectively.

Differential evolutionary (DE) algorithm is an efficient algorithm for solving mixed integer nonlinear programming problem at present. Although the differential evolution algorithm is easy to fall into local optimum, the convergence rate is loved by many scholars in practical applications. Deep[5] improved crossover and mutation operator based on genetic algorithm, and used Deb constraint rules to choose the excellent individual into next generation. Maiti [14] proposed a RCGA algorithm with ranking selection, whole arithmetic crossover and uniform mutation to solve the problem. Lin [12] proposed MIHDE algorithm which contains the migration operation to avoid candidate individuals clustering together. The population diversity measure is introduced to inspect when the migration operation should be performed so that the algorithm can use a smaller population size to obtain a global solution. Kitayama [8] designed the penalty function to convert discrete variable into continuous variable for mixed integer nonlinear programming problems and used particle swarm optimization algorithm to solve the problem. Costa [2] through the experiment demonstrate that the evolutionary algorithm as a valid approach to the optimization of non-linear problems. Mahdavi [13] employs a novel method for generating new solution vectors that enhances accuracy and convergence rate of harmony search (HS) algorithm. Yan [29] proposed an improved line-up competition algorithm and applied this algorithm to handle the mixed integer nonlinear programming problems. Test results show that the algorithm is efficient and robust.

In the previous study, the scholars only use one single algorithm for solving mixed integer programming problems. Single algorithm has its own prominent aspect. But for mixed integer programming problems, there are two types of variables: continuous variables and discrete variables, the single evolutionary algorithm is flawed, so scholars began to study hybrid algorithm.

Hedar [6] transformed the constrained optimization problem into unconstrained bi-objective optimization problem, the Pareto dominance is used for individual evaluation, and combine the pattern search algorithm into genetic algorithm framework to improve the efficiency of the algorithm for solving the problem. Liao [11] studied the ant colony algorithm, proposed three mixed thoughts. local search algorithm and differential evolutionary algorithm were introduced in the ant colony algorithm framework respectively. Final, three algorithm cooperative hybrid enhanced ability of optimization. Srinivas [21] added tabu search in differential evolution algorithm, avoid a lot of duplication search, greatly improving the computational efficiency. Liao [10] presented a hybrid algorithm that include differential evolution algorithm, local search operator and harmony search algorithm. The test results show the hybrid algorithm is effectively for solving the engineering design optimization problems. Yi [30] combined differential evolution algorithm, local search operator, harmony search algorithm and particle swarm optimization to three hybrid algorithm based on the literature [10]. Schluter [19] proposed extended version hybrid algorithm based on ant colony algorithm framework which effectively solve the high-dimensional non-convex and computation complicated optimization problems.

The rest of the paper is organized as follows: in Section 2 some basic information are described, Section 3 describes the original differential evolution algorithm and its variants, Section 4 presents the proposed method, Section 5 presents the experimental results, Section 6 concludes our study. Appendix shows the test problem used in this paper.

2. Background information

In this paper, we using the following method transformed the constrained optimization problem into unconstrained bi-objective optimization problem [2, 5, 6, 8, 12, 13, 14, 29].

$$g_i'(x,y) = (max\{0, g_i(x,y)\})^t, i = 1, ..., m,$$
$$h_j'(x,y) = |h_j(x,y)|^t, j = 1, ..., n,$$

where, usually $t = 1$ or $t = 2$. Using this method formula (1.1) can be converted to the multi-objective optimization problem as follows:

$$\begin{aligned} min \quad & f(x,y), \\ & g_i'(x,y), i = 1, 2, ..., m, \\ & h_j'(x,y), j = 1, 2, ..., n. \end{aligned} \tag{2.1}$$

By defining $f_c = \sum_{i=1}^{m} g_i' + \sum_{j=1}^{n} h_j'$, the formula (2.1) can be converted to the bio-objective optimization problem as follows:

$$\begin{aligned} min \quad & f(x,y), \\ & f_c(x,y). \end{aligned} \tag{2.2}$$

Pareto dominate is the best method to solve the problem above. The optimal solution of a multi-objective optimization problem is a set of optimal solution (largely known as Pareto-optimal solutions), that is not the same as in single-objective optimization. Two basic concept of multi-objective optimization have shown below

(1) Pareto dominate: A decision vector x^0 is said to dominate a decision vector x^1 (also written as $x^0 \prec x^1$) if and only if

$$\forall i \in \{1, ..., m\} : f_i(x^0) \le f_i(x^1),$$
$$\wedge \exists j \in \{1, ..., m\} : f_j(x^0) < f_j(x^1).$$

(2) Pareto optimal set: The Pareto optimal set P_s is defined as $P_s = \{x^0 | \neg \exists x^1 \succ x^0\}$ also called non-dominated optimal set.

3. Differential evolution algorithm

3.1. Basic differential evolution algorithm

Differential evolution (DE) [22, 23, 24] algorithm is a very simple but effective evolutionary algorithm, which is similar to the genetic algorithm. 1995, Storn and Price first proposed "differential evolution" this new concept in technical report [22]. A year later, differential evolution algorithm was successful demonstration at the first session of the International Competition in evolutionary optimization. With the subsequent development, differential evolution algorithm is used in various fields. Swagatam Das [3] made a detail summary for differential evolution algorithm, from the basic concept to core operators, and application in multi-objective, constraints, large-scale, uncertainty optimization problems, which reflects the strong performance of differential evolution algorithm.

Differential evolution algorithm is a swarm intelligence algorithm; its main steps include mutation, crossover and selection which are described briefly in the following.

Mutation operation: For each individual $x_g^m (m = 1, 2, ..., ps)$(where ps is the population size, g is the current generation)in this generation, differential vector is generated by two different individuals x_{r1}^g,x_{r2}^g from the parent generation. The differential vector is defined as $D_{1,2} = x_{r2}^g - x_{r3}^g$. The mutation operation is defined as:

$$v_m^g = x_{r1}^g + F * (x_{r2}^g - x_{r3}^g), \tag{3.1}$$

where $x_{r1}^g = (x_{r_1,1}^g, x_{r_1,2}^g, ..., x_{r_1,D}^g)$, $x_{r2}^g = (x_{r_2,1}^g, x_{r_2,2}^g, ..., x_{r_2,D}^g)$ and $x_{r3}^g = (x_{r_3,1}^g, x_{r_3,2}^g, ..., x_{r_3,D}^g)$ are randomly selected from the parent generation and $r_1 \neq r_2 \neq r_3 \neq m$, $F \in [0, 2]$, which called mutation constant.

Crossover operation: For each mutation individual v_m^g , a trial individual u_m^g is generated, using probability cross operations on the each dimension. The scheme is as follows:

$$u_{mn}^g = \begin{cases} v_{mn}^g & if \quad rand \leq cp \ or \ n = rand\_n, \\ x_{mn}^g & else, \end{cases} \tag{3.2}$$

where $rand$ is a random number generator within [0,1], $cp \in [0, 1]$ is a crossover rate, the value of cp is larger, the contribution of v_m^g to trial individual u_m^g is greater, $rand\_n$ is randomly chosen from {1,2,...,D},which ensures that u_m^g gets at least one element from v_m^g.

Selection operation: The purpose of selection operation is to determine which individual is better between trial individual u_m^g and target individual x_m^g. For minimization problem, substituting the target individual x_m^g with the trial individual u_m^g if the fitness of u_m^g is smaller than the x_m^g. The scheme is as follows

$$x_m^{g+1} = \begin{cases} u_m^g & if \quad f(u_m^g) \leq f(x_m^g), \\ x_m^g & else. \end{cases} \tag{3.3}$$

3.2. Constraints handling method

For optimization problems with constraints, scholars have made a lot of methods to deal with constraints in the optimization problem. The penalty function method is one of the easiest and the earliest constraint handling method. For each infeasible, calculates the fitness and plus a big number which we call penalty constant thereby reducing the selection probability of this individual. Huang [7] designed a special penalty function to handle the constraints in the mixed integer problem. A self-adaptive penalty function was proposed by Tessema [26]. Multi-objective constraint handling method [3, 6, 7, 11, 19, 21, 22, 23, 24, 26, 27, 30] is that handles all the constraints as an objective function, with the original objective function form the bio-objective unconstrained optimization problems. The individual selected into the new population based on the fitness and smaller constraint violation. Deb[4] proposed constrained-domination, the quality of two individual based on the following criteria: if both are infeasible, select the individual who violates less constraints; if both are feasible, using Pareto-dominate to choose the individual; feasible ones are always considered as better than the infeasible ones. Mallipeddi in the literature[15] made a detail review for the constraint handling techniques used with the EAs and proposed a novel constraint handling procedure called ECHT. The experimental results showed that the ECHT outperforms all constraint handling methods, as well as the state-of-the-art methods.

In this paper, we propose a new handling constraint method which called (PF). PF benefits from two methods, multi-objective constraint handling and SF (superiority of feasible solutions). In order to choose a better individual, we use the following method: If the dominate relationship exists, we use multi-objective constrained handling method, and if the two individuals are nondominated with respect to each other, we use SF. The Pseudo-code of the PF is stated as follows.

Function of constraint handling of PF

Input: x denotes parent individual, ff denotes function value of x (size is 1×2), x_{new} denotes offspring individual, ff_{new} denotes function value of x_{new} (size is 1×2)
Output: x_{newg} denotes new generation individual, ff_{newg} denotes function value of x_{newg} (size is 1×2)
1)If $x \prec x_{new}$
2) $x_{newg} \leftarrow x$;$ff_{newg} \leftarrow ff$;
3)Elseif $x \succ x_{new}$
4) $x_{newg} \leftarrow x_{new}$;$ff_{newg} \leftarrow ff_{new}$;
5)Elseif $x == x_{new}$
6) $x_{newg} \leftarrow x_{new}$ or $x_{newg} \leftarrow x$;$ff_{newg} \leftarrow ff$
7)Else
8) If $ff(2) == 0$
9) $x_{newg} \leftarrow x$;$ff_{newg} \leftarrow ff$
10) Elseif $ff_{new}(2) == 0$
11) $x_{newg} \leftarrow x_{new}$;$ff_{newg} \leftarrow ff_{new}$
12) Elseif $ff(2) < ff_{new}(2)$
13) $x_{newg} \leftarrow x$;$ff_{newg} \leftarrow ff$
14) Elseif $ff(2) > ff_{new}(2)$
15) $x_{newg} \leftarrow x_{new}$;$ff_{newg} \leftarrow ff_{new}$;
16) End If
17)End If
18)Return x_{newg} and ff_{newg}

3.3. Discrete variable handling method

Mixed integer problems are optimization problems with some discrete decision variables and continuous variables. Satisfying the integer restrictions are very difficult but important. Current approach to discrete variable is as following:

(1) Truncation procedure: in order to ensure that, after crossover and mutation operations have been performed, the integer restrictions often use the truncation procedure or round to the integer or rounding depending on the probability [5].

(2) This method proposed by [10] involves the following operations: 1) replace each discrete variable by a continuous variable taking values between 1 and n with n being the number of discrete values allowed for the discrete variable being considered. This variable is called a continuous position variable; 2)truncate or round the value assigned to each continuous position to integer; 3)use the integer position value to look up the actual value from the corresponding discrete set.

In this paper, discrete part of the individual adopts integer coding. NP sequences of integer are used to compose the integer part of population, then genetic algorithm takes place to generate the next generation.

4. Differential genetic algorithm co-evolution

4.1. Basic genetic evolution algorithm

Genetic algorithm, as a mature, efficient random search algorithm, is widely used to solve practical problems [5, 6, 14, 16]. In practical applications, there have been many improvements such as different genetic expression, crossover and mutation operators, using special operators, different regeneration and selection methods and so on. In this paper, a co-evolution approach which takes advantage of DE and GA is adopted for the mixed integer problem. Adopted genetic algorithm handles the integer part in each individuals. Designed by the following operations:

(1) Coding techniques. Using integer coding technique, randomly generate NP individuals that meet the requirements in integer set.

(2) Selection. Traditional GA selects two individuals from parent generation to the next crossover operation while DE selects three individuals. So we proposed an improved genetic algorithm with new selection operation which randomly selects three different individuals from parent generation.

(3) Crossover. The crossover operator is applied on the three selected individuals; randomly generates two cross bit to generate a new individual. For example, the crossover operation has illustrated in Fig.1, where the gene fragments must satisfy the integer set {0,1,2,...,9}.

Fig. 1: Crossover operation

(4) Mutation operation. In order to generate trail individual, randomly generate an integer from integer set {0,1,2,...,9} replace the gene on the mutation position. Specific method is as Fig. 2.:

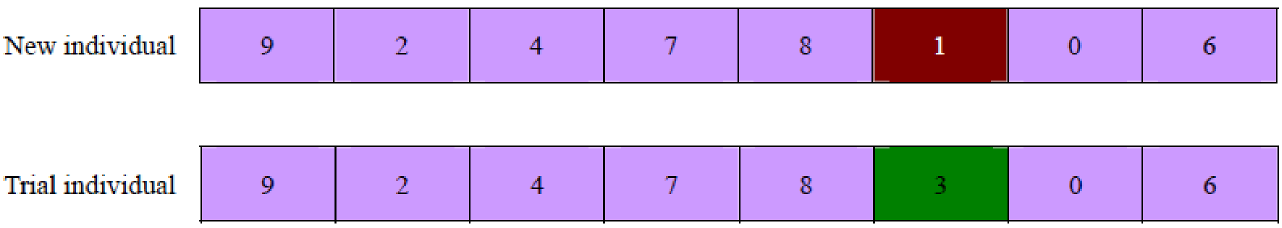

Fig. 2: Mutation operation

4.2. Differential genetic co-evolution algorithm

DE and GA are all very good swarm intelligence optimization algorithms. These two algorithms, basically, have a same framework, but they have their own specific operating characteristics. There are some difficulties which appear in the combination of those two algorithms. So we proposed an improved genetic algorithm as describe in Section 4.1 that is very effectively to solve the problem. Conventional hybrid algorithms are based on two or more serial algorithm that means the population will go through multiple algorithms operator which will exponentially increase the algorithm's time complexity. D-GCE, proposed in this paper, uses a divide-and-conquer idea that means the differential evolution algorithm and genetic algorithm respectively handle the continuous part and discrete part. The flowchart of the D-GCE algorithm is shown in Fig. 3.

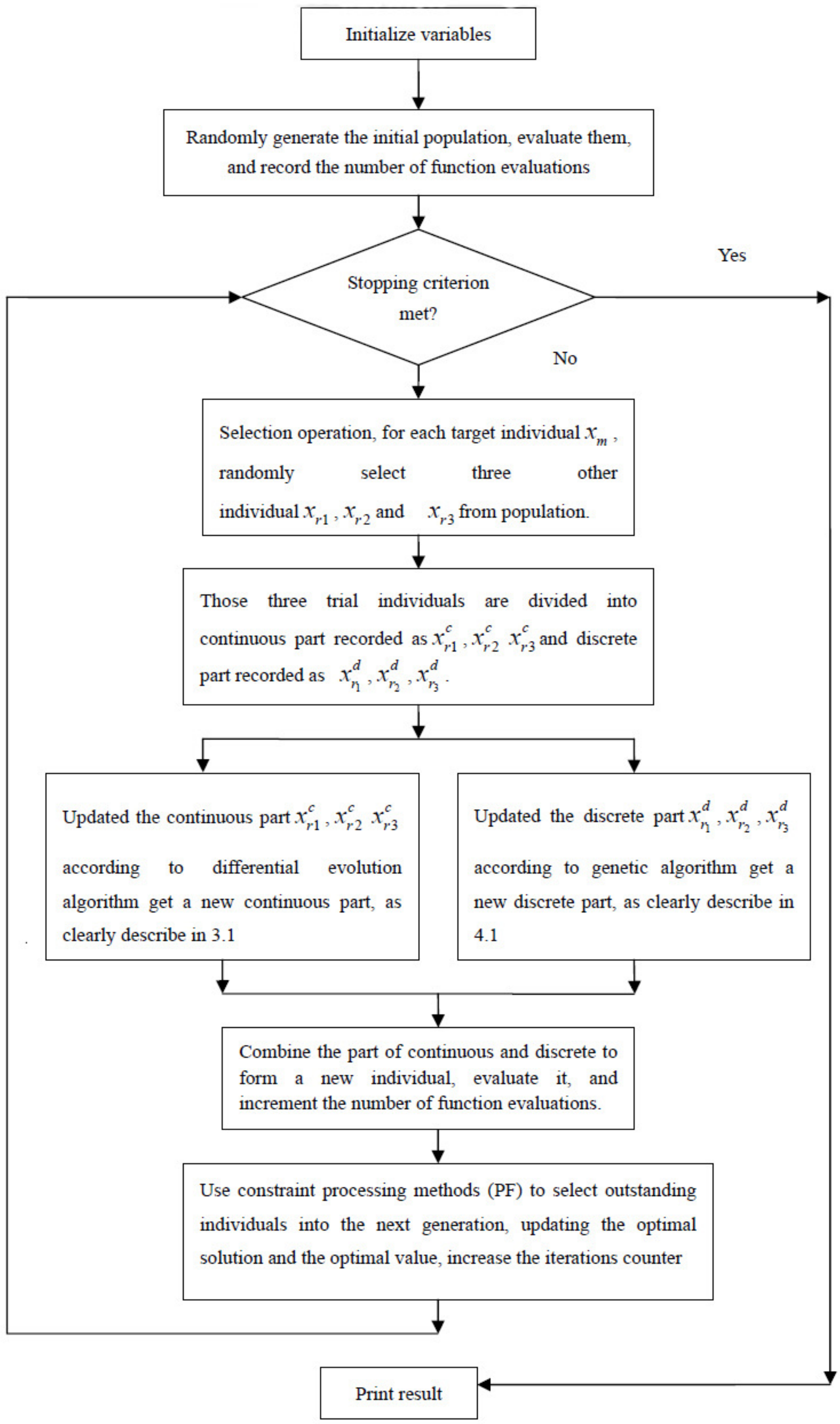

Figure 3: Flowchart of the D-GCE algorithm

The D-GCE algorithm is outlined by pseudo code in the following section to solve the model we proposed.

---

1)Specify DGCE-related parameter values, which include NP (population size), $inMAX$ (max iterations), F (scaling factor), CR (crossover probability) and MR (mutation value)
2)Initial the global optimal $gbest$ of the minimization problem
3)Randomly generate the initial population P in the feasible region
4)Evaluate the initial population, record the number of function evaluations
5)Determine the best solution x_{best}, and the best function value $fbest$
6)Set the current iterations $in = 1$
7)While $in < inMAX$ and $fbest > gbest$
9) For each target vector $x_i (i = 1, 2, ..., NP)$
10) Randomly select three individuals (x_{ri}, x_{r2}andx_{r3}) from population P ,where $i \neq r1 \neq r2 \neq r3$
11) Three individuals are divided into continuous part recorded as x^c_{r1} , x^c_{r2} and x^c_{r3} , and discrete part recorded as x^d_{r1} , x^d_{r2} and x^d_{r3}
12) Update the continuous part x^c_{r1} , x^c_{r2} x^c_{r3} to generate the mutated vector according to Eq.(3.1)
13) Generate the trial vector x_{tri} according to Eq.(3.2).$DP(i, :) = x_{tri}$
14) Update the discrete part x^d_{r1} , x^d_{r2} and x^d_{r3} to generate the new vector according to Fig.1
15) Use new vector to generate the mutated vector x_{mut} according to Fig.2,$GP(i, :) = x_{mut}$
16) End For
17) Combine DP and GP as the new population , $Poff$ and calculate the function values of $Poff$, record the function evaluations
18) Update population according to Pseudo-code 1
19) Update the best solution x_{best}, and the best function value $fbest$
20) Increase iterations in
21)End While

---

5. Experiment and results

In our experiments, 24 mixed-integer test problems [1, 5, 6, 10] are used to investigate the potential of D-GCE. Those test problems are selected from published literature in several different engineering fields. Problems $p.1 - p.14$ taken from literature[10] whereas $p.2 - p.9$ also appears in literature[5, 6] and $p.15 - p.24$ taken from literature [1]. All programs were coded in Matlab and all executions were made on a Dell personal computer with Intel(R)Core(TM) i5-3570K CPU @ 3.40 GHz.

Table 1: Description of test problems

P	N	Nr	Ni	f	fr(%)	ne	le	ni	li
1	4	2	2	Linear	0	0	1	0	2
2	5	2	3	Linear	0	2	0	0	3
3	7	3	4	Nonlinear	25.5231	0	0	4	5
4	2	1	1	Linear	18.2323	0	0	1	1
5	2	1	1	Nonlinear	31.8359	0	0	1	0
6	3	2	1	Quadratic	0.187	0	0	1	2
7	3	2	1	Linear	0	0	0	2	2
8	7	3	4	Nonlinear	15.0124	0	0	4	5
9	5	2	3	Quadratic	100	0	0	3	0
10	10	0	10	Linear	0.0191	0	0	4	0

11	4	0	4	Nonlinear	11.7783	0	0	3	0
12	8	4	4	Nonlinear	13.0680	0	0	3	0
13	4	4	0	Nonlinear	1.4667	2	1	1	0
14	2	1	1	Nonlinear	0.0160	0	0	2	0
15	2	2	0	Linear	44.2316	0	0	2	0
16	6	3	3	Nonlinear	0.1452	0	0	2	0
17	2	0	2	Nonlinear	27.5206	0	0	2	1
18	2	0	2	Quadratic	29.9766	0	0	2	1
19	2	1	1	Nonlinear	37.3432	0	0	1	1
20	2	2	0	Nonlinear	73.1288	0	0	0	2
21	2	1	1	Quadratic	49.9616	0	0	0	1
22	2	2	0	Nonlinear	0.0003	0	0	1	0
23	2	2	0	Nonlinear	62.7279	0	0	2	0
24	6	6	0	Linear	0.7846	0	0	5	0

5.1. Test problems and parameters setting

The basic information of 24 test problems are shown in Table 1 where ne denote the number of non-linear equations, le denotes the number of linear equations, ni denotes the number of non-linear inequalities, li denotes the number of linear-inequalities, N denote the number of decision variables, Nr denote the number of real variables, Ni denotes the number of integer variables. From Table 1 we can see that problem 13, 15, 20, 22, 23 and 24 only have real variables, problem 10, 11, 17 and 18 only have integer variables and other problems have both type of variables. ca denotes the number of active constraints. fr uses the following formula to calculate, denotes the feasible rate of search.

$$fr = NS/C, \tag{5.1}$$

whereand $C = 1000000$ and NS is the number of feasible solutions.

The population size plays a vital role for the performance of intelligent evolutionary algorithm. If the population size is too small, reducing the diversity of the solution, on the contrary will increase the time complexity of the algorithm [9]. Therefore, we must choose an appropriate scale population. Storn [22] proposed an idea that the population size should increase with the dimension of variables. The population size in Wang´s paper [28] equals to the 5-10 times of variable dimension. Mohamed [17] used the following formulate to calculate the population size:

$$NP = \begin{cases} 20*n & 2 \leq n < 5, \\ 10*n & 5 \leq n < 10, \\ 5*n & n \leq 10. \end{cases} \tag{5.2}$$

In this paper, the population size equals to the 5-10 times of variable dimension. The mutation constant F and crossover constant cp can also affecting the search efficiency of the algorithm. Self-adaptive parameter setting is proposed by Brest to improve the ability of the algorithm. In this paper, the mutation constant $F = 0.5$ and crossover constant $cp = 0.3$ is a fixed value in DE. GA used the following parameter values: the crossover constant $Gcp = 0.6$ and mutation constant $Gmp = 0.3$. Some scholars use the tolerable error approach to change the equality constraints into inequality constraints. Different people use different accuracy, such as in Ali´s paper [1] $\delta = 0.01$, Mallipeddi [15] adopt $\delta = 0.0001$, $\delta = 1.0E - 10$ was adopted by Mohamed [17]. In this paper, we set the tolerable error $\delta = 1.0E - 6$.

5.2. Result obtained by the D-GCE algorithm

Table 2: Result obtained by the D-GCE algorithm

P	Optimal	Best	Median	Mean	Worst	Std
1	87.5	87.500013	87.500907	87.495666	87.509925	0.002892
2	7.667	7.667180	7.667180	7.667180	7.667180	0.000000
3	4.5796	4.579582	4.579603	4.579604	5.632729	0.000005
4	2	2.000000	2.000000	2.000001	2.000001	0.000000
5	2.1247	2.124470	2.124592	2.124662	2.125538	0.000094
6	1.076543	1.076543	1.076543	1.076544	1.076544	0.000000
7	99.245209	99.239554	99.239554	99.239554	99.239555	0.000000
8	3.557463	3.557461	3.557463	3.569764	4.632729	0.107520
9	-32217.4	-32217.427780	-32217.427780	-32217.427780	-32217.427780	0.000000
10	-0.808844	-0.808844	-0.808844	-0.806037	-0.790126	0.006717
11	-0.974565	-0.974565	-0.974565	-0.974222	-0.972759	0.000697
12	-0.999486	-0.999901	-0.999607	-0.999649	-0.999487	0.000108
13	5850.770	5848.122647	6090.526202	6185.328999	6771.596851	218.993775
14	-75.1341	-75.134167	-75.134054	-75.134053	-75.134000	0.000043
15	-5.50796	-5.508010	-5.507984	-5.507985	-5.507970	0.000009
16	-316.27	-316.531661	-305.836287	-298.581657	-216.639945	23.553381
17	0.18301	0.183015	0.183015	0.183015	0.183015	0.000000
18	0	0	0	0	0	0
19	-195.37	-195.370581	-195.370284	-195.370243	-195.367739	0.000296
20	-2.2137	-2.213662	-2.213661	-2.213661	-2.213660	0.000000
21	0.125	0.125000	0.125000	0.125000	0.125000	0.000000
22	0.0821	0.082085	0.082092	0.082092	0.082100	0.000004
23	1.5087	1.508653	1.508691	1.508687	1.508710	0.000016
24	-0.388811	-0.777376	-0.484066	-0.504627	-0.391925	0.089488

Table 3: The result of equality constraint problem using the algorithm D-GCE

	$p.1$	$p.2$	$p.3$
	12.499998140749165	1.1180339887498895	37.9114274237522
	0.000002220001002	1.310370697104448	238.4555666064445
$[x, y]$	1	0	0.7316905492784
	0	1	0.3616750176226
		1	
$h_1(x, y)$	0.000000000000005	2.220446049250313e-016	2.398081733190338e-014
$h_2(x, y)$		-4.440892098500626e-016	-1.110223024625157e-014
$f(x, y)$	87.500001420800686	7.667180068813135	5848.122647741006

To avoid the randomness, all the test problems independently ran 100 times. Table 1 shows the result of 24 test problems using the algorithm D-GCE. For all problems, the algorithm D-GCE can converge to the global optimal except problem 3, 8,10,11,13 and 16. Problem 3 and 8 not all reach the global optimal value in 100 times (about 5 times). For problem 10 and 11, the algorithm D-GCE is very close to the global optimal value. Problem 13 and 16 are most difficult with relatively high standard and the best solution is far away from the global optimal. But consider about problem 2, 4, 6, 7, 9, 17, 18, 20 and 21, those problems have perfect result and the Std =0 and problem 3, 5, 14, 15, 22 and 23 obtain relatively low Std. For all problems, the algorithm D-GCE found the global optimal within error, especially in problem 1, 5, 7, 8, 12, 13, 16 and 24, the best solution obtained by D-GCE algorithm are superior to the known global optimal. Table 3 show the results of the test problems contain equality constraints using the algorithm D-GCE. From the Table 3, we can see that D-GCE obtains the solution with very small error compared with global optimal in all test problems. In summary, we believe that the D-GCE algorithm is an effective approach to handle mixed integer constrained optimization problem.

It is worth mentioning that there are some errors in the literature [10] (problem 3, problem7 and problem 12) when doing numerical experiments. correct results and test functions are shown in Appendix.

Fig.4 shows the convergence curve of test problems 1-24 (except p.13), where the horizontal axis represents the number of iteration t, ordinate represents $\log(f(x) - f(x^*))$, x denotes the best solution in t generation and x^* is the known best solution. From the Fig.4, we can clearly see that $\log(f(x) - f(x^*))$ decreases with the increase of iterations, that means the solution obtained by D-GCE algorithm is getting close to the known best solution. Table 4 shows that $\log(x)$ is corresponding different precision x which can reflect the convergence degree of the test problems. The convergence curve of problem 1, 2 and 20 indicate that $\log(f(x) - f(x^*)) < -10$, it means $f(x) - f(x^*) > 0.00001$. The result of rest test problems is close to the known best solution, i.e. $f(x) - f(x^*)$ close to 0.000001. It is worth noting problem 13 is difficult to obtain the best solution. The algorithm D-GCE with a few iterations converge to the optimal solution in problem 2, 9, 12, 18 and 24.

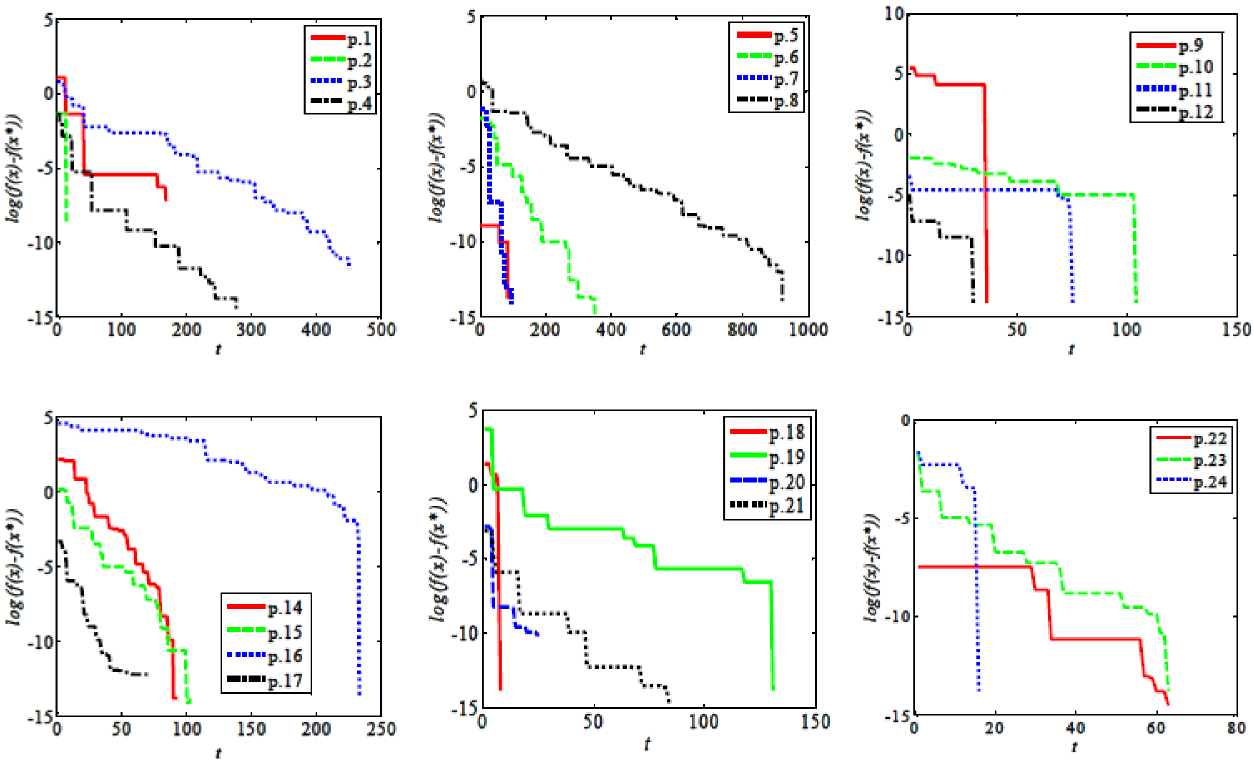

Figure 4: Curve of the variation of $\log(f(x) - f(x^*))$ with generation t

Table 4: The variation of $\log(x)$ with different precision x

x	0.01	0.001	0.0001	0.00001	0.000001	0.0000001
$\log(x)$	-4.6052	-6.9078	-9.2103	-11.5129	-13.8155	-16.1181

5.3. Comparison results of D-GCE and algorithms from [10]

In order to detection how competitive of the proposed approach was, it was compared with the algorithms selected from [10]. The experiments use five indicators to measure the performance of the algorithm, including the best value, the success rate, the number of function evaluations, mean and standard deviation, the results have shown in Table 5. Each indicator of the algorithm D-GCE is calculated through 100 runs. Comparing the results presented in Table 5, notice that the success rate of test problems over 80% especially the success rate of problem 1, 2, 4-7, 9, 12 and 14 reaches 100% use the algorithm D-GCE. In case of problem 13, the success rate is 10%. mde´-his[1]is one of the best three algorithm that are from [10], but compared with the algorithm proposed by this paper, the success rate of problem 10, 11 and 13 are better than the algorithm D-GCE. Overall the success rate of the proposed algorithm is better than mde´-his, ma-mde´ and mde. Consider about the number of function evaluations, problem 1, 2, 6 and 12 using D-GCE lower than other three algorithms and with 100% success rate. The number of function evaluations and the success rate using D-GCE in problem 7 and 8, higher than other three algorithms. We use a higher number of function evaluations exchange a good result, which is in line with ¨ no free lunch in the world ¨. Problems 11 and 13 are most difficult with relatively low success rate and high number of function evaluation. The best solution can not reflect the overall performance of the algorithm, so we calculate the average of solution. From Table 5, we can see that the Mean indicator of problem 1,2,6,7 and 12 using D-GCE is better than other three algorithms. The Mean indicator of problem 13 is far away from the optimal.

Table 5: Comparison result of D-GCE, mde´-his, ma-mde´ and mde´

P	Optimal	**Best**	Indicators	**D-GCE**	mde´-his[1]	ma-mde´[1]	mde´[1]
			Success rate	**1**	1	0.867	0.533
1	87.5	**87.428783**	Number of evaluations	**1993**	5359	4463	7777
			Mean	**87.495666**	87.497550	88.230145	89.879034
			Standard	**0.018077**	0.002118	1.899683	2.768746
			Success rate	**1**	0.2	0.067	0.033
2	4.5796	**4.579582**	Number of evaluations	**1666**	83442	93524	96718
			Mean	**7.667180**	7.848896	7.883841	7.918619
			Standard	**0.000000**	0.121909	0.098982	0.047891
			Success rate	**0.97**	0.8	1	0.933
3	7.667	**7.667180**	Number of evaluations	**15717**	14518	13023	7688
			Mean	**4.579604**	4.579599	4.579595	4.661414
			Standard	**0.000005**	0.000005	0.000003	0.311365
			Success rate	**1**	1	1	0.933
4	2	**2.000000**	Number of evaluations	**3704**	3297	1430	1075
			Mean	**2.000001**	2.000001	2.000000	2.009348
			Standard	**0.000000**	0.000000	0.000000	0.043579
			Success rate	**1**	1	1	0.9
5	2.1247	**2.124470**	Number of evaluations	**1294**	1409	653	827

			Mean	**2.124662**	2.124604	2.124574	2.167894
			Standard	**0.000094**	0.000076	0.000071	0.132196
			Success rate	**1**	0.833	0.533	0.4
6	1.076543	**1.076543**	Number of evaluations	**12416**	22146	25766	30986
			Mean	**1.076544**	1.094994	1.099805	1.1244531
			Standard	**0.000000**	0.052898	0.055618	0.075163
			Success rate	**1**	0.967	1	1
7	99.239554	**99.239554**	Number of evaluations	**1958**	449	684	403
			Mean	**99.240933**	99.241271	99.512250	99.239554
			Standard	**0.001429**	0.001842	1.485279	0.000000
			Success rate	**0.95**	0.833	0.867	0.3
8	3.557463	**3.557461**	Number of evaluations	**43792**	27116	20116	37739
			Mean	**3.569764**	3.561157	3.564912	3.599903
			Standard	**0.107520**	0.008381	0.029017	0.059012
			Success rate	**1**	1	1	1
9	-32217.4	**-32217 .427780**	Number of evaluations	**609**	493	1955	1240
			Mean	**-32217 .427262**	-32217 .427780	-32217 .427106	-32217 .427262
			Standard	**0.000000**	0.000000	0.003690	0.002836
			Success rate	**0.85**	1	0.9	0.933
10	-0.808844	**-0.808844**	Number of evaluations	**7964**	9733	35180	21593
			Mean	**-0.806037**	-0.808844	-0.807907	-0.807608
			Standard	**0.006717**	0.000000	0.003077	0.005615
			Success rate	**0.82**	1	0.933	0.967
11	-0.974565	**-0.974565**	Number of evaluations	**5887**	176	449	333
			Mean	**-0.974222**	-0.974565	-0.974335	-0.974505
			Standard	**0.000697**	0.000000	0.000977	0.000330
			Success rate	**1**	1	1	1
12	-0.999486	**-0.999486**	Number of evaluations	**769**	1800	2742	1669
			Mean	**-0.999649**	-0.999635	-0.999638	-0.999631
			Standard	**0.000108**	0.000104	0.000111	0.000104
			Success rate	**0.1**	0.267	0.233	0.067
13	5850.770	**5850.770**	Number of evaluations	**54907**	38237	39902	46868
			Mean	**6175 .328999**	6082 .551078	6040 .005940	6070 .604982
			Standard	**218.993775**	185.056741	168.603518	109.163780
			Success rate	**1**	1	1	1
14	-75.1341	**-75.1341**	Number of evaluations	**3412**	4266	2827	1679
			Mean	**-75.134053**	-75.134137	-75.134130	-75.134137
			Standard	**0.000043**	0.000025	0.000024	0.000023

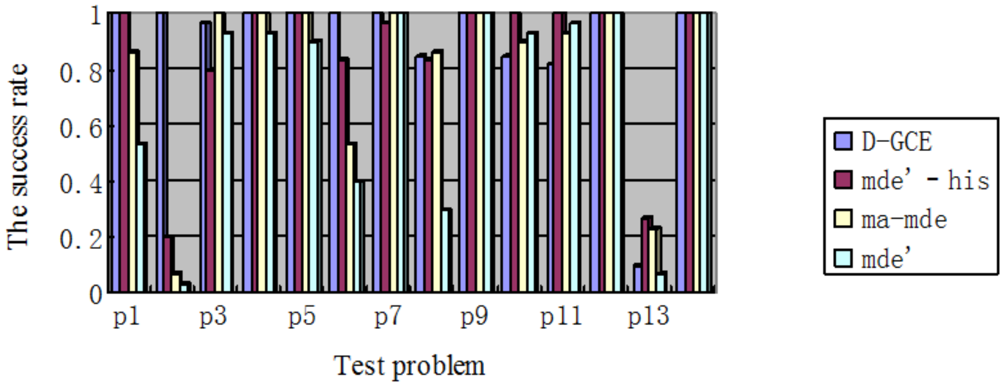

Figure 5: Comparison results of D-GCE, mde´-his, ma-mde´ and mde´ based on success rate

5.4. Comparison results of D-GCE and algorithms from [30]

Four indicators, the best solution, the success rate, the number of function evaluations and the CPU time, are used to reflect the performance of D-GCE, MDE´-HJ, MDE´-IHS-HJ and PSO-MDE´-HJ. From Table 6, the value of success rate that use D-GCE algorithm outperforms the other three algorithms except problem 11and 13. Considering about the CPU time indicator, the result of problem 1, 2, 5, 6, 8, 9, 10, 12 and 14 completely superior to other three algorithms. Problem 11and 13 are still difficult to obtain the global optimal with the worst number of evaluations when compared with other three algorithm, in problem 1, 2, 6, 8, 10 and 12, D-GCE obtains competitive results compared to MDE´-HJ, MDE´-IHS-HJ and PSO-MDE´-HJ.

Table 6: Comparison result of D-GCE, MDE´-HJ, MDE´-IHS-HJ and PSO-MDE´-HJ

P	Optimal	**Best**	Indicators	**D-GCE**	MDE´-HJ [16]	MDE´ -IHS-HJ [16]	PSO-MDE´-HJ [16]
1	87.5	**87.428783**	Success rate	**1**	1	0.96	1
			Number of evaluations	**1993**	5859	6589	4596
			CPU time	**0.18**	0.58	0.74	0.33
2	7.667	**7.667180**	Success rate	**1**	0.74	0.94	0.73
			Number of evaluations	**1666**	28389	10522	20910
			CPU time	**0.024**	4.03	1.47	1.70
3	4.5796	**4.579582**	Success rate	**0.97**	0	0.48	0
			Number of evaluations	**15717**	15795	15116	15511
			CPU time	**1.34**	1.8	0.9	1.27
4	2	**2.000000**	Success rate	**1**	0.99	0.95	0.88
			Number of evaluations	**3704**	1787	1211	1863
			CPU time	**0.33**	0.28	0.16	0.19
5	2.1247	**2.124470**	Success rate	**1**	0.79	0.86	0.49
			Number of evaluations	**1294**	1721	1251	2776
			CPU time	**0.081**	0.3	0.18	0.27
6	1.076543	**1.076543**	Success rate	**1**	0.9	0.83	0.71
			Number of evaluations	**12416**	15964	21890	22969

			CPU time	**0.7**	2.42	2.71	2.25
			Success rate	**1**	0.91	0.99	1
7	99.239554	**99.239554**	Number of evaluations	**1446**	994	458	412
			CPU time	**0.14**	0.17	0.05	0.04
			Success rate	**0.93**	0.03	0.81	0.28
8	3.557463	**3.557461**	Number of evaluations	**38522**	50210	45821	49206
			CPU time	**3.65**	5.84	4.01	3.63
			Success rate	**1**	1	1	1
9	-32217.4	**-32217.427780**	Number of evaluations	**609**	495	453	555
			CPU time	**0.04**	0.07	0.07	0.05
			Success rate	**0.85**	0.47	0.92	0.89
10	-0.808844	**-0.808844**	Number of evaluations	**7964**	43090	13152	24484
			CPU time	**0.66**	9.95	2.49	3.31
			Success rate	**0.82**	1	1	1
11	-0.974565	**-0.974565**	Number of evaluations	**5887**	285	221	288
			CPU time	**0.43**	0.09	0.07	0.06
			Success rate	**1**	1	1	1
12	-0.999486	**-0.999486**	Number of evaluations	**796**	1704	1762	1414
			CPU time	**0.06**	0.22	0.26	0.15
			Success rate	**0.1**	0.76	0.5	0.99
13	5850.770	**5850.770**	Number of evaluations	**54907**	30138	32618	18265
			CPU time	**4.03**	4.51	3.60	1.90
			Success rate	**1**	1	1	1
14	-75.1341	**-75.1341**	Number of evaluations	**3412**	3058	1747	2419
			CPU time	**0.16**	0.48	0.26	0.23

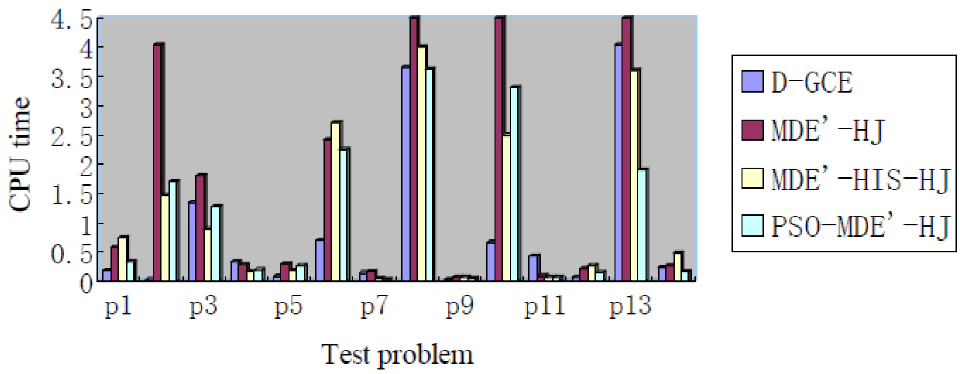

Figure 6: Comparison result of D-GCE, MDE´-HJ, MDE´-IHS-HJ and PSO-MDE´-HJ based on CPU time

5.5. Comparison results of D-GCE and algorithms from [11]

Liao proposed three hybrid algorithms base on Ant Colony Algorithm [11] and presents the success rate and CPU time for 14 test problems. The comparison results of D-GCE and other four algorithms based on the success rate and CPU time were shown in Table 7. For all problems, the D-GCE algorithm found the global optimal in at least one run, whereas problems 2,3and 8 did not found the optimal solution using the ACOR algorithm, and the algorithm ACO_R-DE did not find the optimal solution of problem 2. In case of 9 problems, D-GCE algorithm provides 100% success. Moreover, only in 1 problem its success rate is less than 60%. In case of ACO_R algorithm, 100% success rate is achieved in 5 problems, but in 6 problems success rate is less than 60%. In case of ACO_R-HJ algorithm, 100% success rate is achieved in 5 problems but in 5 problems success rate is less than 60%. In case of ACO_R-DE algorithm, 100% success rate is achieved in 6 problems, but in 2 problems success rate is less than 60%. Consider ACO_R-DE-HJ algorithm, 100% success rate is achieved in 5 problems but in 6 problems success rate is less than 60%. D-GCE algorithm also obtained completely less CPU time than ACO_R , ACO_R-HJ, ACO_R-DE and ACO_R-DE-HJ algorithm in problem 1,2,5,6,9,10 and 12. Overall, D-GCE, proposed by this paper, is superior to the algorithm proposed by literature [11].

Table 7: Comparison result of textbfD-GCE, ACO_R , ACO_R-HJ, ACO_R-DE and ACO_R-DE-HJ algorithm

P	D-GCE		ACO_R [14]		ACO_R-HJ [14]		ACO_R-DE [14]		ACO_R-DE-HJ [14]	
	Success rate	t-CPU	Success rate	t-CPU	Success rate	t-CPU	Success rate	t-CPU	Success rate	t-CPU
1	**1**	**0.18**	0.0667	5.6485	0.7	0.516	0.8333	1.648	0.7	0.656
2	**1**	**0.024**	0	33.625	0.8667	0.4135	0	39.406	0.2333	5.2425
3	**0.97**	**1.34**	0	7.4375	0.9333	0.8125	1.0	4.5625	0.9333	0.75
4	**1**	**0.33**	0.9	0.2575	0.9	0.1645	1.0	0.188	1.0	0.141
5	**1**	**0.081**	0.6	0.1795	0.4	0.219	0.8667	0.109	0.8667	0.094
6	**1**	**0.7**	0.0333	11.469	0.0333	2.86	0.5333	12.492	0.5	3.0
7	**1**	**0.14**	1.0	0.164	1.0	0.0545	0.8333	0.141	1.0	0.0545
8	**0.93**	**3.65**	0	22.969	0.1333	3.532	0.9667	8.2345	0.0667	3.531
9	**1**	**0.04**	1.0	0.047	1.0	0.047	1.0	0.078	1.0	0.047
10	**0.85**	**0.66**	1.0	10.7025	0.1	5.39	1.0	8.5705	0.3333	5.0235
11	**0.82**	**0.43**	1.0	0.141	1.0	0.047	0.9667	0.102	1.0	0.047
12	**1**	**0.06**	0.3333	7.047	1.0	0.367	1.0	1.4765	0.0333	1.578
13	**0.1**	**4.03**	0.9	6.7885	0.0667	5.5465	0.9	2.5855	0.0667	5.344
14	**1**	**0.16**	1.0	0.781	1.0	0.172	1.0	1.0395	1.0	0.203
Total	**12.67**	**11.8250**	7.8333	107.2570	9.1333	20.1415	11.9	80.633	8.7333	25.7115

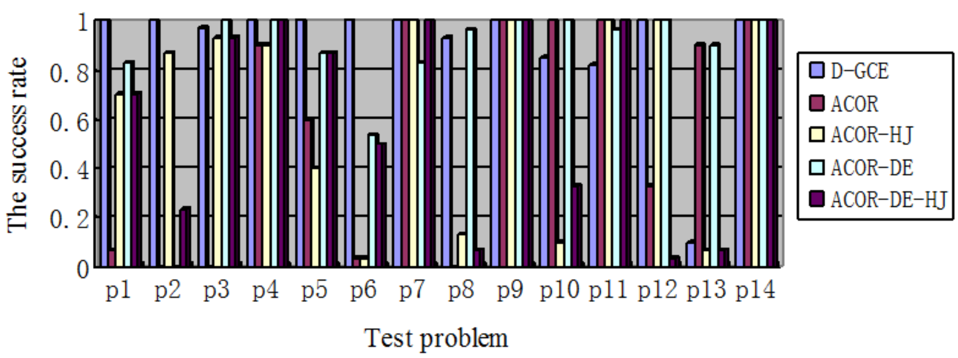

Figure 7: Comparison results of D-GCE,ACO_R , ACO_R-HJ, ACO_R-DE and ACO_R-DE-HJ algorithm based on CPU time

5.6. Comparison results of D-GCE and algorithms from [1]

In order to get a better insight into the relative performance of D-GCE and the algorithm proposed by [1], choose the test problems $p.15 - p.24$, six performance indicators is calculated which including the optimal, the best, the mean, the standard, the average number of iterations and the average number of evaluations in respect of those algorithms. For all problems, the D-GCE algorithm found the global optimal in at least one run, especially in problem 16 and 24 the solution is superior to the global optimal. For problems 15, 17-24, the best and mean is very close to the optimal, whereas the standard of problem 17, 18, 20 and 21 is close to 0 under the preset accuracy. D-GCE algorithm required less number of iterations than other four algorithms in all problems except problem 16 and 21. Consider about the number of evaluations, D-GCE algorithm is also superior to all other algorithms in all problems, the visual comparison can be found in the histograms 8 and 9.

Table 8: Comparison results of D-GCE, GA_SFP, GA_PFP, LEDE_SFP and LEDE_PFP algorithm

P	**Optimal**	Indicators	**D-GCE**	GA_SFP	GA_PFP	EDE_SFP	LEDE_PFP
		Best	**-5.508010**	-5.51	-5.51	-5.51	-5.51
		Mean	**-5.507985**	-5.51	-5.51	-5.51	-5.51
15	**-5.50796**	Standard	**0.000009**	0.0	0.0	0.00	0.00
		The average number of iterations	**92.56**	110.6	110.9	NA	NA
		The average number of evaluations	**2806.8**	NA	NA	11614.4	11593.45
		Best	**-316.531661**	-314.03	-314.391	-316.27	-316.27
		Mean	**-298.581657**	-295.43	-295.65	-310.05	-307.8
16	**-316.27**	Standard	**23.553381**	13.41	12.12	17.76	20.26
		The average number of iterations	**777.32**	313.1	319.3	NA	NA
		The average number of evaluations	**23319.6**	NA	NA	26029.7	27137.01
		Best	**0.183015**	0.18	0.18	0.18	0.18
		Mean	**0.183015**	0.18	0.18	0.18	0.18
17	**0.18301**	Standard	**0.000000**	0.00	0.00	0	0
		The average number of iterations	**68.02**	101.3	101.3	NA	NA
		The average number of evaluations	**2070**	NA	NA	10377.07	10404.72
		Best	**0**	0	0	0	0
		Mean	**0**	0	0	0	0
18	**0**	Standard	**0**	0.00	0.00	0	0
		The average number of iterations	**60**	101.8	101.6	NA	NA
		The average number of evaluations	**1567.2**	NA	NA	10481.89	10504.99
		Best	**-195.370581**	-195.37	-195.37	-195.37	-195.37
		Mean	**-195.370243**	-195.37	-195.37	-195.37	-195.37
19	**-195.37**	Standard	**0.000296**	0.01	0.00	0	0
		The average number of iterations	**114.15**	123	121.6	NA	NA
		The average number of evaluations	**2303**	NA	NA	13157.11	13252.961
		Best	**-2.213662**	-2.21	-2.21	-2.21	-2.21
		Mean	**-2.213661**	-2.21	-2.21	-2.21	-2.21
20	**-2.2137**	Standard	**0.000000**	0.00	0.00	0	0

		The average number of iterations	**32.1**	100.9	101	NA	NA
		The average number of evaluations	**616.8**	NA	NA	10331.63	10314.72
		Best	**0.125000**	0.13	0.13	0.13	0.13
		Mean	**0.125000**	0.13	0.13	0.13	0.13
21	**0.125**	Standard	**0.000000**	0.00	0.00	0	0
		The average number of iterations	**282**	101.1	101.1	NA	NA
		The average number of evaluations	**8496**	NA	NA	10357.18	10314.72
		Best	**0.082085**	0.08	0.08	0.08	0.08
		Mean	**0.082092**	0.08	0.08	0.08	0.08
22	**0.0821**	Standard	**0.000004**	0.00	0.00	0.00	0.00
		The average number of iterations	**41.3**	103.86	104.75	NA	NA
		The average number of evaluations	**1582.9**	NA	NA	10958.83	11033.13
		Best	**1.508653**	1.51	1.51	1.51	1.51
		Mean	**1.508687**	1.51	1.51	1.51	1.51
23	**1.5087**	Standard	**0.000016**	0.00	0.00	0.00	0.oo
		The average number of iterations	**53**	102.97	103.10	NA	NA
		The average number of evaluations	**1941**	NA	NA	10700.91	10687.7
		Best	**-0.777376**	-0.41	-0.41	-0.41	-0.41
		Mean	**-0.504627**	-0.41	-0.41	-0.41	-0.41
24	**-0.388811**	Standard	**0.089488**	0.01	0	0.00	0.00
		The average number of iterations	**73**	134.3	136	NA	NA
		The average number of evaluations	**2763.2**	NA	NA	12922.71	12941.93

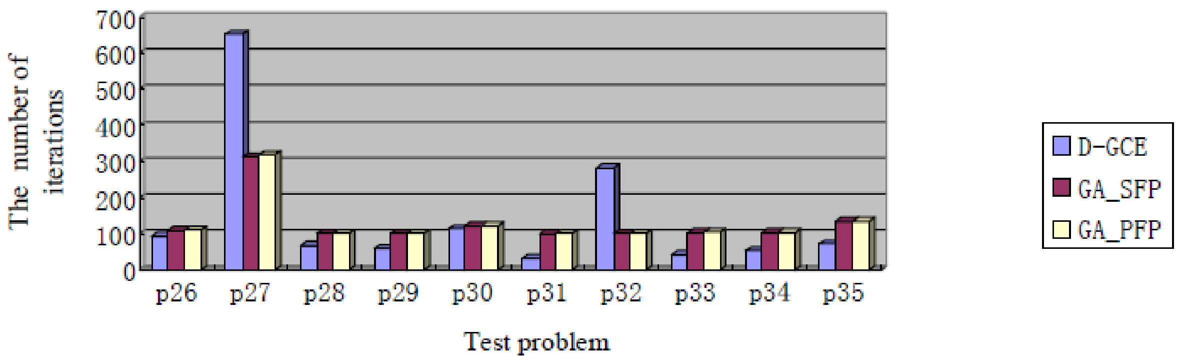

Figure 8: Comparison result of D-GCE, GA_SFP and GA_PFP based on the number of iterations

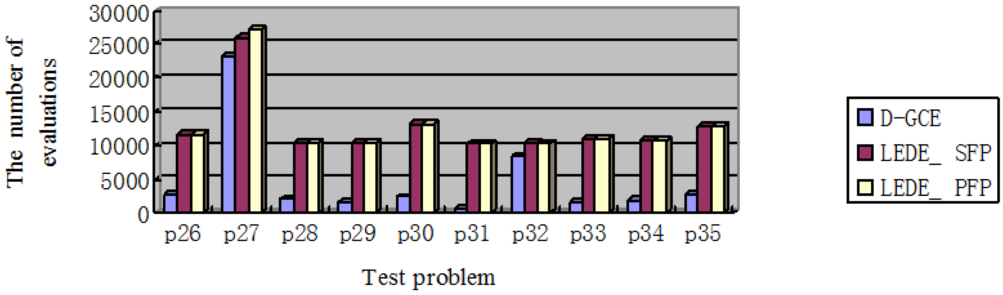

Figure 9: Comparison result of D-GCE, LEDE_SFP and LEDE_PFP based on the number of evaluations

6. Conclusion

In this paper, a differential genetic co-evolutionary algorithm is proposed for solving of constrained, integer and mixed integer optimization problems. In this algorithm we use differential evolution algorithm to handle the continuous part and the genetic algorithm to handle the discrete integer part in each individual. In order to handle the constraints in the model, we proposed a new method which we called PF is used to handle the constraints of the optimization problems.

The performance of the proposed D-GCE algorithm is compared with some classic algorithm selected from literature on a set of 24 test problems. Our results show that the proposed D-GCE algorithm outperforms other algorithm in most of the case for solving nonlinear mixed integer programming problem. Especially in some indicators such as the success rate, the efficiency of the search process, the quality of the solution and the stability of algorithm, etc. These properties can reflect that the D-GCE algorithm is an effective, stable, competitive evolutionary algorithm. However, this article only focuses on the parallel of those two algorithms and the control parameters, and self-adaptive would be needed to study about a stronger algorithm. In addition, in this paper, we used a new method to convert the constrained mixed integer problems to unconstrained bio-objective optimization problem and achieved better results, so we will extend this idea to the field of multi-objective optimization.

Appendix

P.1

$min \quad F = 6.4x_1 + 6x_2 + 7.5y_1 + 5.5y_2$

$s.t. \quad 0.8x_1 + 0.67x_2 = 10, x_1 - 20y_1 \leq 0, x_2 - 20y_2 \leq 0$

$\quad x_1, x_2 \in [0, 2], y_1, y_2 \in \{0, 1\}$

The known global optimal solution is $F^* = 87.5$, $x = [12.5006, 0]$ and $y = [1, 0]$

P.2

$min \quad F = 2x_1 + 3x_2 + 1.5y_1 + 2y_2 - 0.5y_3$

$s.t. \quad x_1^2 + y_1 = 1.25,\ x_2^{1.5} + 1.5y_2 = 3,\ x_1 + y_1 \leq 1.6$

$\quad 1.333x_2 + y_2 \leq 3, -y_1 - y_2 + y_3 \leq 0$

$\quad x_1, x_2 \in [0, 2], y_1, y_2, y_3 \in \{0, 1\}$

The known global optimal solution is $F^* = 7.667$, $x = [1.118, 1.310]$ and $y = [0, 1, 1]$

P.3

$min \quad (x_1 - 1)^2 + (x_2 - 2)^2 + (x_3 - 3)^2 + (y_1 - 1)^2 + (y_2 - 2)^2 + (y_3 - 1)^2 - ln(y_4 + 1)$

$s.t. \quad x_1 + x_2 + x_3 + y_1 + y_2 + y_3 \leq 5,\ x_1^2 + x_2^2 + x_3^2 + y_3^2 \leq 5.5,\ x_1 + y_1 \leq 1.2$

$\quad x_2 + y_2 \leq 1.8,\ x_3 + y_3 \leq 2.5,\ x_1 + y_4 \leq 1.2,\ x_2^2 + y_2^2 \leq 1.64,\ x_3^2 + y_3^2 \leq 4.25,\ x_3^2 + y_2^2 \leq 4.64$

$\quad x_1 \in [0, 1.2], x_2 \in [0, 1.28], x_3 \in [0, 2.062], y_1, y_2, y_3, y_4 \in \{0, 1\}$

The known global optimal solution is $F^* = 4.5796$, $x = [0.2, 0.8, 1.908]$ and $y = [1, 1, 0, 1]$

$P.4$

$$\begin{aligned} min \quad & F = 2x + y \\ s.t. \quad & 1.25 - x^2 - y \leq 0, x + y \leq 1.6 \\ & x \in [0, 1.6], y \in \{0, 1\} \end{aligned}$$

The known global optimal solution is $F^* = 2$, $x = [0.5]$ and $y = [1]$

$P.5$

$$\begin{aligned} min \quad & F = -y + 2x_1 - ln(x_1/2) \\ s.t. \quad & -x_1 - ln(x_1/2) + y \leq 0 \\ & x_1 \in [0.5, 1.4], y \in \{0, 1\} \end{aligned}$$

The known global optimal solution is $F^* = 2.1247$, $x = [1.375]$ and $y = [1]$

$P.6$

$$\begin{aligned} min \quad & F = -0.7y + 5(x_1 - 0.5)^2 + 0.8 \\ s.t. \quad & -exp(x_1 - 0.2) - x_2 \leq 0, x_2 + 1.1y \leq -1.0, x_1 - 1.2y \leq 1.2 \\ & x_1 \in [0.2, 1], x_2 \in [-2.22554, -1], y \in \{0, 1\} \end{aligned}$$

The known global optimal solution is $F^* = 1.076543$, $x = [0.94194, -2.1]$ and $y = [1]$

$P.7$

$$\begin{aligned} min \quad & F = 7.5y + 5.5(1 - y) + 7x_1 + 6x_2 + \frac{50(1 - y)}{0.8[1 - exp(-0.4x_2)]} + \frac{50y}{0.9[1 - exp(-0.5x_1)]} \\ s.t. \quad & 0.9[1 - exp(-0.5x_1)] - 2y \leq 0, 0.8[1 - exp(-0.4x_2)] - 2(1 - y) \leq 0, x_1 \leq 10y, x_2 \leq 10(1 - y) \\ & x_1, x_2 \in [0, 10], y \in \{0, 1\} \end{aligned}$$

The known global optimal solution is $F^* = 99.245209$, $x = [3.514237, 0]$ and $y = [1]$

$P.8$

$$\begin{aligned} min \quad & (x_1 - 1)^2 + (x_2 - 2)^2 + (x_3 - 3)^2 + (y_1 - 1)^2 + (y_2 - 1)^2 + (y_3 - 1)^2 - ln(y_4 + 1) \\ s.t. \quad & x_1 + x_2 + x_3 + y_1 + y_2 + y_3 \leq 5, x_1^2 + x_2^2 + x_3^2 + y_3^2 \leq 5.5, x_1 + y_1 \leq 1.2 \\ & x_2 + y_2 \leq 1.8, x_3 + y_3 \leq 2.5, x_1 + y_4 \leq 1.2, x_2^2 + y_2^2 \leq 1.64, x_3^2 + y_3^2 \leq 4.25, x_3^2 + y_2^2 \leq 4.64 \\ & x_1 \in [0, 1.2], x_2 \in [0, 1.8], x_3 \in [0, 2.5], y_1, y_2, y_3, y_4 \in \{0, 1\} \end{aligned}$$

The known global optimal solution is $F^* = 3.557463$, $x = [0.2, 1.28062, 1.95448]$ and $y = [1, 0, 0, 1]$

$P.9$

$$\begin{aligned} min \quad & F = 5.357854x_1^2 + 0.835689y_1x_3 + 37.29329y_1 - 40795.141 \\ s.t. \quad & 85.334407 + 0.0056858y_2x_3 + 0.0006262y_1x_2 - 0.0022053x_1x_3 \leq 92, \\ & 80.51249 + 0.0071317y_2x_3 + 0.0029955y_1y_2 + 0.0021813x_1^2 - 90 \leq 20 \\ & 9.300961 + 0.0047026x_1x_3 + 0.0012547y_1x_1 + 0.0019085x_1x_2 - 20 \leq 5 \\ & x_1, x_2, x_3 \in [27, 45], y_1 \in \{78, ..., 102\}, y_2 \in \{33, ..., 45\} \end{aligned}$$

The known global optimal solution is $F^* = -32217.4$, $x = [27, any, 27]$ and $y = [78, any]$

$P.10$

$$\begin{aligned} min \quad & F = -\prod_{j=1}^{10}[1 - (1 - p_j)^{y_j}] \\ s.t. \quad & \prod_{j=1}^{10}(a_{ij}y_j^2 + c_{ij}y_j) \leq b, i = 1, 2, 3, 4 \\ & [p_j] = (0.81, 0.93, 0.92, 0.96, 0.99, 0.89, 0.85, 0.83, 0.94, 0.92) \\ & [a_{ij}] = \begin{bmatrix} 2 & 7 & 3 & 0 & 5 & 6 & 9 & 4 & 8 & 1 \\ 4 & 9 & 2 & 7 & 1 & 0 & 8 & 3 & 5 & 6 \\ 5 & 1 & 7 & 4 & 3 & 6 & 0 & 9 & 8 & 2 \\ 8 & 3 & 5 & 6 & 9 & 7 & 2 & 4 & 0 & 1 \end{bmatrix} \end{aligned}$$

$$[c_{ij}] = \begin{bmatrix} 7 & 1 & 4 & 6 & 8 & 2 & 5 & 9 & 3 & 3 \\ 4 & 6 & 5 & 7 & 2 & 6 & 9 & 1 & 0 & 8 \\ 1 & 10 & 3 & 5 & 4 & 7 & 8 & 9 & 4 & 6 \\ 2 & 3 & 2 & 5 & 7 & 8 & 6 & 10 & 9 & 1 \end{bmatrix}$$

$[b_i] = (2.0 \times 10^{13}, 3.1 \times 10^{12}, 5.7 \times 10^{13}, 9.3 \times 10^{12})$

$y_j \in \{1, ..., 6\}, j = 1, ..., 10$

The known global optimal solution is $F^* = -0.808844$ and $y = [2, 2, 2, 1, 1, 2, 3, 2, 1, 2]$

$P.11$

$min \quad F = -\prod_{j=1}^{4} R_j$

$s.t. \quad \sum_{j=1}^{4} d_{1j} y_j^2 \leq 100, \sum_{j=1}^{4} d_{2j}(y_j + exp(y_j/4)) \leq 150, \sum_{j=1}^{4} d_{3j} y_j exp(y_j/4) \leq 160,$

$y_j \in \{1, ..., 6\}, j = 1, 2, 4 y_3 \in \{1, ..., 5\}$

$where\ R_1 = 1 - q_1((1 - \beta_1)q_1 + \beta_1)_1^y - 1, R_2 = 1 - (\beta_2 q_2 + p_2 q_2^{y_2})/(p_2 + \beta_2 q_2)$

$R_3 = 1 - q_3^{y3}, R_4 = 1 - q_4((1 - \beta)q_4 + \beta_4)^{y_4 - 1}$

$[p_j] = (0.93, 0.92, 0.94, 0.91), [q_j] = (0.07, 0.08, 0.06, 0.09)$

$[\beta_j] = (0.2, 0.06, 0.0, 0.3)$

$$[d_{ij}] = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 7 & 7 & 5 & 7 \\ 7 & 8 & 8 & 6 \end{bmatrix}$$

The known global optimal solution is $F^* = -0.974565$ and $y = [3, 3, 2, 3]$

$P.12$

$min \quad F = -\prod_{j=1}^{4}[1 - (1 - x_j)^{y_j}]$

$s.t. \quad \sum_{j=1}^{4} v_j y_j^2 \leq 250,$

$\sum_{j=1}^{4} \alpha_j (\frac{-100}{ln(x_j)})^{\beta_j}(y_j + exp(y_j/4)) \leq 400,$

$\sum_{j=1}^{4} w_j * y_j * exp(y_j/4) \leq 500,$

$x_j \in [0.5, 1 - 10^{-6}], j = 1, 2, 3, 4 \quad y_j \in \{1, ..., 10\}, j = 1, 2, 3, 4$

$where\ [v_j] = (1, 2, 3, 2), [w_j] = (6, 6, 8, 7), [\alpha_j] = (1.0, 2.3, 0.3, 2.3) \times 10^{-5}$

$[\beta_j] = (1.5, 1.5, 1.5, 1.5)$

The known global optimal solution is $F^* = -32217.4$, $x = [27, any, 27]$ and $y = [78, any]$

$P.13$

$min \quad F = 0.6224 x_1 x_2 x_3 + 1.7781 x_1^2 x_4 + 3.1661 x_2 x_3^2 + 19.84 x_1 x_3^2$

$s.t. \quad 0.0193 x_1/x_3 - 1 = 0, 0.00954 x_1/x_4 - 1 = 0, x_2/240 - 1 = 0,$

$(1296000 - (4/3)\pi x_1^3)/(\pi x_1^2 x_2) - 1 \leq 0,$

$x_1 \in [25, 150], x_2 \in [25, 240], x_3, x_4 \in [0.0625, 0.125, 1.1875, 1.25],$

The known global optimal solution is $F^* = 5850.770$ and $x = [38.858, 221.402, 0.75, 0.375]$

$P.14$

$min \quad F = -x_1 x_2$

$s.t. \quad 0.145 x_2^{0.1939} x_1^{0.7071} y^{-0.2343} \leq 0.3, 29.67 x_2^{0.4167} x_1^{-0.8333} \leq 7,$

$x_1 \in [8.6, 13.4], x_2 \in [5, 30], y \in \{120, 140, 170, 200, 230, 270, 325, 400, 500\},$

The known global optimal solution is $F^* = -75.1341$ and $x = [13.4, 5.6070]$

$P.15$

$min \quad F = -x_1 - x_2$

$s.t. \quad -8x_1^2 + 8_1^3 - 2x_1^4 + x_2 \leq 2, 96x_1 - 88x_1^2 + 32x_1^3 - 4x_1^4 + x_2 \leq 36,$

$x_1 \in [0, 3], x_2 \in [0, 4]$

The known global optimal solution is $F^* = -5.50796$ and $x = [2.3295, 3.17846]$

$P.16$

$min \quad F = -(0.0204 + 0.0607x_5^2)x_1x_4(x_1 + x_2 + x_3) - (0.0187 + 0.0437x_6^2)x_2x_3(x_1 + 1.57x_2 + x_4)$

$s.t. \quad 2070/x_1x_2x_3x_4x_5x_6 - 1 \leq 0, 6.2x_1x_4x_5^2(x_1 + x_2 + x_3) + 5.8x_2x_3x_6^2(x_1 + 1.57x_2 + x_4) \leq 10000,$

$\quad x_1, x_2 \in [0, 10], x_3, x_4 \in [0, 15], x_5, x_4 \in [0, 1],$

The known global optimal solution is $F^* = -613.27$ and $x = [10, 10, 15, 4.609, 0.78511, 0.3814]$

$P.17$

$min \quad F = \sum_{i=1}^{5} 1/[a_i(x - p_i)(x - p_i) + c_i]$

$s.t. \quad x_1 + x_2 - 5 \leq 0, 6.x_1 - x_2^2 \leq 0, 5x_1^3 + 1.6x_2^2 \leq 0,$

$\quad x_1 \in \{-3, -2, ..., 10\}, x_2 \in \{-4, -3, ..., 7\}$

i	a_i	p_i		c_i
1	0.5	0	5	0.125
2	0.25	2	5	0.25
3	1	3	2	0.1
4	1/12	4	4	0.2
5	2	5	2	1/12

The known global optimal solution is $F^* = 0.18301$ and $x = [-3, -4]^T$

$P.18$

$min \quad F = x_1^2 + x_2^2$

$s.t. \quad x_1 + x_2 - 2 \leq 0, x_1^2 - x_2 \leq 0,$

$\quad x_1 \in \{-3, -2, ..., 2\}, x_2 \in \{0, 1, ..., 5\},$

The known global optimal solution is $F^* = 0$ and $x = [0, 0]$

$P.19$

$min \quad F = -(x_2 - 1.275x_1^2 + 5x_1 - 6)^2 - 10(1 - 1/8\pi)cos(\pi x_1) - 10$

$s.t. \quad -\pi x_1 - x_2 \leq 0, -\pi^2 x_1^2 + 4x_2 \leq 0,$

$\quad x_1 \in [-1.5, 3.5], x_2 \in [0, 15]$

The known global optimal solution is $F^* = -195.37$ and $x = [2.4656, 15]$

$P.20$

$min \quad F = -2x_1 - 6x_2 + x_1^3 + 8x_2^2$

$s.t. \quad x_1 + 6x_2 - 6 \leq 0, 5x_1 + 4x_2 - 10 \leq 0,$

$\quad x_1 \in [0, 2], x_2 \in [0, 1]$

The known global optimal solution is $F^* = -2.2137$ and $x = [0.8165, 0.375]$

$P.21$

$min \quad F = (x_1 - 0.75)^2 + (0.5x_2 - 0.75)^2$

$s.t. \quad x_1 + 0.5x_2 - 1 \leq 0,$

$\quad x_1 \in [0, 1], x_2 \in [0, 2]$

The known global optimal solution is $F^* = 0.125$ and $x = [0.5, 1]$

$P.22$

$min \quad F = exp(x_1 - 2x_2)$

$s.t. \quad sin(-x_1 + x_2 - 1) \leq 0,$

$\quad x_1 \in [-2, 2], x_2 \in [-1.5, 1.5]$

The known global optimal solution is $F^{*}=0.0821$ and $x=[0.5,1.5]$

$P.23$
$min \quad F=x_1\sqrt{1+x_2^2}$
$s.t. \quad 0.124\sqrt{1+x_2^2}\times(8/x_1+1/x_1x_2)-1\leq 0, 0.124\sqrt{1+x_2^2}\times(8/x_1-1/x_1x_2)-1\leq 0,$
$x_1\in[0.2,4], x_2\in[0.1,1.6]$
The known global optimal solution is $F^{*}=1.5087$ and $x=[1.41163,0.377072]$

$P.24$
$min \quad F=-x_4$
$s.t. \quad 0.09755988x_1x_5+x_1-1\leq 0, 0.09658428x_2x_6+x_2-x_1\leq 0, \sqrt{x_5}+\sqrt{x_6}-4\leq 0,$
$0.0391908x_3x_5+x_3+x_1-1\leq 0, 0.03527172x_4x_6+x_4-x_1+x_2-x_3\leq 0$
$x_1,x_2,x_3,x_4\in[0,1], x_,x_6\in[0.00001,16]$
The known global optimal solution is $F^{*}=-0.388811$,
$x=[0.771516,0.516992,0.204192,0.388811,3.03557,5.09726]$

Acknowledgements

This work was supported by the National Natural Science Foundation of P. R. China (61561001) and the Foundations of research projects of State Ethnic Affairs Commission of P. R. China (14BFZ003) and the Foundations of research projects of Beifang University of Nationalities (2015KJ10).

References

[1] M. M. Ali, Z. Kajee-Bagdadi, *A local exploration-based differential evolution algorithm for constrained global optimization*, Appl. Math. Comput., **208** (2009), 31–48. 5, 5.1, 5.6

[2] L. Costa, P. Oliveira, *Evolutionary algorithms approach to the solution of mixed integer non-linear programming problems*, Comput. Chem. Eng., **25** (2001), 257–266. 1, 2

[3] S. Das, *Differential Evolution: A Survey of the State-of-the-Art*, IEEE Trans. Evol. Comput., **15** (2011), 4–31. 3.1, 3.2

[4] K. Deb, *An efficient constraint handing method for genetic algorithms*, Comput. Method Appl. M, **186** (2000), 311–338. 3.2

[5] K. Deep, K. P. Singh, M. L. Kansal, C. Mohan, *A real coded genetic algorithm for solving integer and mixed integer optimization problems*, Appl. Math. Comput., **212** (2009), 505–518. 1, 2, 3.3, 4.1, 5

[6] A. Hedar, *Filter-based genetic algorithm for mixed variable programming*, Numer. Algebra Control Optim., **1** (2011), 99–116. 1, 2, 3.2, 4.1, 5

[7] F. Huang, L. Wang, Q. He, *An effective co-evolutionary differential evolution for constrained optimization*, Appl. Math. Comput., **186** (2007), 340–356. 3.2

[8] S. Kitayama, K. Yasuda, *A method for mixed integer programming problems by particle swarm optimization*, Electr. Eng. Jpn., **157** (2006), 40–49. 1, 2

[9] J. Lampinen, I. Zelinka, *On stagnation of the differential evolution algorithm*, 6th International Mendel Conference on Soft Computing, **2000** (2000), 76–83. 5.1

[10] T. W. Liao, *Two hybrid differential evolution algorithms for engineering design optimization*, Appl. Soft Comput., **10** (2010), 1188–1199. 1, 1, 3.3, 5, 5.2, 5.3

[11] T. W. Liao, R. J. Kuo, J. T. L. Hu, *Hybrid ant colony optimization algorithms for mixed discrete-Ccontinuous optimization problems*, Appl. Math. Comput., **219** (2012), 3241–3252. 1, 3.2, 5.5

[12] Y. C. Lin, K. S Hwang, F. S Wang, *A Mixed-Coding Scheme of Evolutionary Algorithms to Solve Mixed-Integer Nonlinear Programming Problems*, Comput. Math. Appl., **47** (2004), 1295–1307. 1, 2

[13] M. Mahdavi, M. Fesanghary, E. Damangir, *An improved harmony search algorithm for solving optimization problems*, Appl. Math. Comput., **188** (2007), 1567–1579. 1, 2

[14] A. K. Maiti, A. K. Bhunia, M. Maiti, *An application of real-coded genetic algorithm for mixed integer non-linear programming in two-storage multi-item inventory model with discount policy*, Appl. Math. Comput., **183** (2006), 903–915. 1, 2, 4.1

[15] R. Mallipeddi, P. N. Suganthan, *Ensemble of Constraint Handling Techniques*, Evolutionary Comput., **14** (2010), 561–579. 3.2, 5.1

[16] K. Miettinen, M. Makela, J. Toivanen, *Numerical comparison of some penalty based constraint handling techniques in genetic algorithm*, J. Global Optim., **27** (2003), 427–446. 4.1

[17] A. W. Mohamed, H. Z. Sabry, *Constrained optimization based on modified differential evolution algorithm*, Inf. Sci., **194** (2012), 171–208. 5.1, 5.1

[18] L. Pallottino, E. M. Feron, A. Bicchi, *Conflict resolution problems for air traffic management systems solved with mixed integer programming*, IEEE Trans. Intell. Transp. Syst., **3** (2002), 3–11. 1

[19] M. Schluter, J. A. Egea, J. R Banga, *Extended ant colony optimization for non-convex mixed integer nonlinear programming*, Comput. Oper. Res., **36** (2009), 2217–2229. 1, 3.2

[20] T. Schouwenaars, B. De Moor, E. Feron, J. How, *Mixed integer programming for multi-vehicle path planning*, Eur. Control Conf., **1** (2001), 2603–2608. 1

[21] M. Srinivas, G. P. Rangaiah, *Differential evolution with tabu list for solving nonlinear and mixed-integer nonlinear programming problems*, Ind. Eng. Chem. Res., **46** (2007), 7126–7135. 1, 3.2

[22] R. Storn, K. Price, *Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces*, Int. Comput. Sci. Inst., Berkeley, (1995). 3.1, 3.2, 5.1

[23] R. Storn, K. Price, *Minimizing the real functions of the ICEC 1996 contest by Differential Evolution*, IEEE Conference on Evolutionary Computation, **1996** (1996), 3 pages. 3.1, 3.2

[24] R. Storn, K. Price, *Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces*, J. Global Optim., **11** (1997), 341–359. 3.1, 3.2

[25] S. S. Syam, *A dual ascent method for the portfolio selection problem with multiple constraints and linked proposals*, Eur. J. Oper. Res. **108** (1998), 196–207. 1

[26] B. Tessema, G. G. Yen, *A Self Adaptive Penalty Function Based Algorithm for Constrained Optimization*, Evol. Comput., **2006** (2006), 8 pages. 3.2

[27] Y. Wang, Z. Cai, G. Guo, Y. Zhou, *Multiobjective optimization and hybrid evolutionary algorithm to solve constrained optimization problems*, IEEE Trans. Syst. Man Cybern. Part B Cybern., **37** (2007), 560–575. 3.2

[28] L. Wang, L. Li, *An effective differential evolution with level comparison for constrained engineering design*, Struc. Multidisciplinary Optim., **41** (2010), 947–963. 5.1

[29] L. Yan, K. Shen, S. Hu, *Solving mixed integer nonlinear programming problems with line-up competition algorithm*, Comput. Chem. Eng., **28** (2004), 2647–2657. 1, 2

[30] H. Yi, Q. Duan, T. W. Liao, *Three improved hybrid metaheuristic algorithms for engineering design optimization*, Appl. Soft Comput., **13** (2013), 2433–2444. 1, 3.2, 5.4

[31] M. Yuceer, L. Atasoy, R. Berber, *A semiheuristic MINLP algorithm for production scheduling*, Comput. Aided Chem. Eng., **14** (2003), 335–340. 1