http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 10 (11): 2068-2075, 2011
ISSN 1812-5638 / DOI: 10.3923/itj.2011.2068.2075

Modified Differential Evolution Algorithm of Constraimed Nonlinear Mixed Integer Programming Problems

Yuelin Gao, Zaimin Ren and Yang Gao
Institute of Information and System Science, Beifang University for Nationalities,
Yinchuan Ningxia, 750021, China

Abstract: A modified differential evolution algorithm of constrained nonlinear mixed integer programming problems is given. In this algorithm, the positions of the variation particles are self-adaptively adjusted so that the particles evolve in better direction and the feasible basis rule and the dynamic constraint handling technology are added to improve particles'optimization ability. Numerical results show that the proposed algorithm is of high precision and stability.

Key words: Constrained nonlinear mixed integer programming, differential evolution algorithm, constraint handling mechanism

INTRODUCTION

We consider mixed integer programming problem:

$$(\text{MIP})\begin{cases}\min & f(x,y)\\ \text{s.t.} & g_i(x,y)\le 0, \quad i=1,2,\cdots,q\\ & h_j(x,y)=0, \quad j=q+1,q+2,\cdots,m\\ & x^L\le x=(x_1,x_2,\cdots,x_{n_c})\le x^U\\ & y^L\le y=(y_1,y_2,\cdots,y_{n_I})\le y^U\end{cases} \tag{1}$$

where $(x, y) = (x_1, x_2, ..., x_{nc}, y_1, y_2, ..., x_{nI})$ is the vector of solutions such that $x \in S_1 \subseteq R^{nc}$ and $y \in S_1 \subseteq I^{nI}$, q is the number of inequality constraints. The search space S_1 and S_2 were defined as an n_c-dimensional and n_I-dimensional space bounded by parametric constraints:

$$x_d^L \le x_d \le x_d^U, d=1,2,\cdots,n_C$$
$$y_d^L \le y_d \le y_d^U, d=1,2,\cdots,n_I$$

Mixed integer programming (Costa and Oliveira, 2001; Lin *et al*., 2004) is recognized as a class of NP complete problems, solving integer programming problem is not an easy task. Many scholars present different approaches to solve mixed integer programming problems. These can be divided into two categories: Deterministic algorithm and evolutionary algorithm. For example, branch and bound method (Nowak and Vigerske, 2008), Generalized Benders decomposition, Outer approximation method Bergamini *et al*. (2008) Dantzig-Wolf decomposition method Vanderbeck and Savelsbergh (2006) Cutting plane method and their variants, etc. (Lin *et al*., 2004) proposed a mixed-coding scheme of evolutionary algorithms to solve mixed-integer nonlinear programming problems. Solving mixed integer programming with hybrid evolutionary algorithm is proposed by Li *et al*. (2008). Genetic algorithm for non-linear mixed integer programming problems and its applications are proposed by Yokota *et al*. (1996). Chiou and Wang (1998) proposed a hybrid method of differential evolution with applications to optimal control problems of a bioprocess system (Chiou and Wang, 1998). Lin *et al*. (2000) proposed plant scheduling and planning using mixed-integer hybrid differential evolution with multiplier updating.

In this study, it is with a differential evolution algorithm for solving constrained nonlinear mixed integer programming problem. In the proposed modified differential evolution algorithm.

THE BASIC DIFFERENTIAL EVOLUTION ALGORITHM

Differential Evolution (DE) algorithm (Storn and Price, 1995, 1997) proposed for solving polynomials, a common encoding used in the continuous space vector floating-point random search algorithm. The principle of DE is simple, controlled by few parameters, to implement the randomized, parallel and direct global search and easy to understand and implement. In recent years, differential evolution has attracted much attention. In many areas it has been widely applied and has become an important

Corresponding Author: Yuelin Gao, Institute of Information and System Science, Beifang University for Nationalities, Yinchuan Ningxia, 750021, China

branch of evolutionary algorithms. Differential Evolution (DE) with the Genetic Algorithm (GA), has a population variability, cross and selection. But it is different from other algorithms. DE, on the basis of randomly selected parent individuals, generates individual variations. In a certain probability, it crosses the parent individuals with generated individual variations and generates test individuals. Then it uses the greedy strategy to select the individual with better fitness as the progeny of individual between the parent individual and the test individual. Here, are the four basic evolutionary differential evolution algorithm operations.

Operation 1 population initialization: Population initialization of the differential evolution algorithm is the same as other evolutionary algorithms. In the initial population, a design variable of individuals is the floating-point of the uniform random distribution within the boundaries of the upper and lower.

Operation 2 mutation: The mutation of differential evolution is on the basis of the difference vector of the parent individuals. The current evolution of the individual is located x_i^t, i is the current serial number of individuals in the population, t is the evolution generation, randomly selected three individuals from the current population x_{r1}^t, x_{r2}^t and x_{r3}^t (r1 ≠ r2 ≠ r3 ≠ i), taking the difference between two individuals vectors (x_{r1}^t, x_{r2}^t), added to the r1 individual vectors x_{r1}^t after increased by the scaling factor F, get the individual after variance u_i^{t+1}:

$$u_i^{t+1} = x_{r1}^t + F(x_{r2}^t - x_{r3}^t) \tag{2}$$

where, F ∈ [0, 2] is the scaling factor, used to control the zoom level of differential variables.

Operation 3 crossover: The individual after variation u_i^{t+1} and the individual of the current evolution in the populations x_i^t to crossover operation by discrete cross generate individual experiments v_i^{t+1}, the j-th component of v_i^{t+1} is expressed as:

$$v_{ij}^{t+1} = \begin{cases} u_{ij}^{t+1}, & \text{if (rand (0, 1) < CR) 或 (j = randi (1, D))} \\ x_{ij}^t, & \text{other wise} \end{cases} \tag{3}$$

where, rand (0, 1) is random number uniformly distributed in the range (0, 1), randi (1, D) is an integer randomly selected in {1, 2, ..., D}, used to ensure in v_i^{t+1} at least one is by the u_i^{t+1} contribution, CR ∈ [0, 1] is crossover probability, used to control individual trial v_i^{t+1} what are the variables from the individual variation u_i^{t+1} provide which the individual provided by the current evolution of the individual v_i^{t+1}, CR the larger contribution the greater Individual variations u_i^{t+1} to individual of the cross v_i^{t+1}, when CR = 1, $v_i^{t+1} = u_i^{t+1}$.

Operation 4 selection: Greedy selection strategy is used to choose between the parent individual x_i^t and the pilot individual v_i^{t+1}. For the minimization problem, if the fitness value of v_i^{t+1} is smaller than x_i^t, then v_i^{t+1} will replace x_i^t and turn into new individuals of the next generation in the population, otherwise, x_i^t will be retained to the next. Select options to be expressed by:

$$x_i^{t+1} = \begin{cases} v_i^{t+1}, & \text{if } \varphi(v_i^{t+1}) < \varphi(x_i^t) \\ x_i^t, & \text{otherwise} \end{cases} \tag{4}$$

After initialization, through the above variation, crossover and selection operation, cycles operate each individual on the population, get the next generation population, so for several generations and obtain the optimal solution of optimization problems.

MODIFIED DIFFERENTIAL EVOLUTION ALGORITHM

Population initialization: Suppose the dimension of the real variable is n_c, the dimension of the integer variables is n_i, the i-th particles (x_i, y_i) can be expressed as (x_{i1}, x_{i2}, ..., x_{inc}, y_{i1}, y_{i2}, ..., y_{inI}) and the real variables to initialize according to the formula (5):

$$x_i^0 = x^L + \rho(x^U - x^L) \tag{5}$$

$$y_i^0 = \lfloor y^L + \rho(y^U - y^L) \rfloor \tag{6}$$

where, ρ is random number uniformly distributed in the range (0, 1), $\lfloor * \rfloor$ is the integer by rounding.

For the integer variables, first, according to the formula (6), get a value randomly in the real space, then use the following rounding rules (Deep *et al*., 2009) and get the corresponding integer value of the variable $\bar{y}_{ij}$.

(1) If $y_{ij} \geq \lfloor y_{ij} \rfloor + 0.5$, then $\bar{y}_{ij} = \lfloor y_{ij} \rfloor + 1$

(2) If $y_{ij} < \lfloor y_{ij} \rfloor + 0.5$, then $\bar{y}_{ij} = \lfloor y_{ij} \rfloor$

where, $[y_{ij}]$ is an maximum integer that does not exceed number y_{ij}. This initial population is selected to ensure the randomness of the particles, the uniformity of the distribution.

- Mutation

The current evolution individual is located (x_i^t, y_i^t), the current global optimal solution is (gbx, gby). Take linear vector (rx_i^t+(1-r) gbx, ry_i^t+(1-r) gby) composed by the current evolution individual (x_i^t, y_i^t) and the current global optimal solution (gbx, gby). Randomly select two individuals (x_{r1}^t, y_{r1}^t) (x_{r2}^t, y_{r2}^t) (r1 ≠ r2 ≠ i) from the current population. Take the difference between the two individual vectors (x_{r1}^t-x_{r2}^t) (y_{r1}^t-y_{r2}^t). After being increased by the scaling factor F and added to the linear vector, get the variance individual (x_u, y_u).

Use Eq. 7 to cause variability of real variables and use Eq. 8 the variation and then use rounding rules of the above on the integer variable:

$$x_u = rx_i^t + (1-r)gbx + F(x_{r1}^t - x_{r2}^t) \tag{7}$$

$$y_u = \lfloor ry_i^t + (1-r)gby + F(y_{r1}^t - y_{r2}^t) \rfloor \tag{8}$$

$$r = 1 - t / Tmax$$

where, T_{max} is the largest number of iteration, t is the current number of iterations, $\lfloor * \rfloor$ is the integer by rounding.

- Crossover

Crossover operation can maintain the population diversity. The specific equation is:

$$x_v = \begin{cases} x_{vj}, & \text{if (rand (0, 1) < CR) (j = randi (1, D))} \\ x_{ij}^t, & \text{other wise} \end{cases} \quad j = 1, 2, ..., n, \tag{9}$$

$$y_v = \begin{cases} y_{vj}, & \text{if (rand (0, 1) < CR) (j = randi (1, D))} \\ y_{ij}^t, & \text{other wise} \end{cases} \quad j = 1, 2, ..., n, \tag{10}$$

where, rand (0, 1) is random number uniformly distributed in the range (0, 1), CR ∈ [0, 1], randi (1, D) is an integer randomly selected in {1,. 2, ..., D}.

In order to balance the global search ability and local search ability well, this study uses the number of iterations index increased with the probability factor of the cross, updating formula (Deng, 2008) as follows:

$$CR = CR_{min} + (CR_{max} - CR_{min}) * \exp(-a * (1 - t / T_{max}) \wedge b)$$

where, a = 30, b = 3, CR_{min} = 0.1, CR_{max} = 0.9.

- Selection

The update equation of the select operation:

$$(x_i^{t+1}, y_i^{t+1}) = \begin{cases} (x_v, y_v), & \text{if } \varphi(x_v, y_v) < \varphi(x_i^t, y_i^t) \\ (x_i^t, y_i^t), & \text{otherwise} \end{cases} \tag{11}$$

Greedy selection strategy is used between the parent individual (x_i^t, y_i^t) and the pilot individual (x_v, y_v) to choose operation, where, φ (x) representative of the fitness function, choose the individual whose fitness is the optimum (for the minimization problem, the fitness function value is smaller individuals).

- Constraint handling mechanism

With the constraints in the solution of nonlinear mixed integer programming problems, constraint handling is critical. Here, constraint violation (Lu and Chen, 2008) of the particle x is defined as:

$$G(x) = \sum_{i=1}^{m} \max\{0, g_i(x)\} + \sum_{j=1}^{n} \max\{0, |h_j(x)| - \delta\} \tag{12}$$

where, δ is a small positive number. G (x) is the sum of constraint violations and G (x)≥0 and thus G (x) = 0 for x is in the range of the feasible region. Moreover, all the optimal solutions of G (x) = 0 constitute the feasible region of the original problem.

A feasible basis rules: Two individual solutions are compared in the following rules (Deb, 2000):

Solution 1: A feasible solution is always preferred to an infeasible one.

Solution 2: Between two feasible solutions, the one having better objective function is preferred.

Solution 3: Between two infeasible solutions, the one having smaller constraint violation is preferred.

B Dynamic-objective method for constraint-handling Constraint violation G (x) by defining.The original MIP is converted into the following unconstrained bi-objective optimization problem min (G (x), f (x)), G (x) is considered the first objective, f (x) is considered the second objective. G (x) is used to determine whether each particle in the feasible region. It can be seen Eq. 12 no parameter. In view of this case, in order to be close to the feasible region, each particle has the ability to dynamically adjust its objectives f (x) or G (x) according to the following

method: if one particle is outside the feasible region, then G (x) is as the objective function to optimize; otherwise f (x) is as the objective function to optimize. It should be noted that after the particle starts optimizing f (x), it is still likely for this particle to fly out of the feasible region. In this situation, the particle would give up optimizing f (x) and turn again to minimize G (x). The purpose of doing so is to make particles into the feasible region but not into the strict control of particles in the feasible region, thus improving the search ability of particles. This dynamically adjusts its objectives algorithm (Lu and Chen, 2006) as follows:

$$\text{set } G_{gbest} = G(gbx), gbf = f(gbx)$$

$$\text{If } G_i < G_{gbest} \text{ then } gbx = x_i^t, G_{gbest} = G_i, gbf = f_i, \text{ End}$$

$$\text{If } G_i = 0 \text{ and } G_{gbest} = 0$$

$$\text{If } f_i <= gbf, \text{ then } gbx = x_i^t, gbf = f_i, \text{End}$$

where, G_{gbest} is a constraint violation of the global best particle, gbf is the global optimal solution.

The algorithm description:

Step 1: Set the maximum number of iterations and initialize the parameters;
Step 2: According to Eq 5 and 6 random initialization of each individual and calculate the objective function and constraint violation;
Step 3: According to the feasible basis rules,find the global minimum gbf of the initial particle swarm, the global optimal solution gbx and the constraint violation G_{gbest} of the global optimal particles;
While (t<Tmax) does.
Step 4: Randomly select two individuals that are different from (x_i^t, y_i^t) in the population, according to Eq. 7, 8 mutation, Eq. 9, 10 crossover, generating test individual (x_y, y_y);
Step 5: According to Eq. 11, to select and produce t+1 generations of individual (x_i^{t+1}, y_i^{t+1});
Step 6: Calculate each new particle's objective function and constraint violation; according to dynamic-objective method for constraint-handling, update the global optimal gbf, the global optimal solution gbx and constraint violation G_{gbest} of the global optimal particles;
t = t+1;
End while
Output gbf and gbx.

EXPERIMENTS AND DISCUSSIONS

It selects 14 well-known benchmark functions g 01 ~ g 14 to test the performance of the new algorithm. They are standard functions with a constrained nonlinear integer programming problem. You can fully test the performance of new algorithms, function g 01 ~ g 13 from g 01 ~ g 13 of [12] and function g 14 from g 15 of [12], where g 09 and g 10 are maximization problems, the use of -f (x) is to be translated into the minimize problem, the rest of the test functions are minimization problem.

Numerical experiments are completed in the software Matlab 7.8. In our experiments, the size of the swarm N = 20, maximum iterations number T_{max} = 800, mutation probability CR_{max} = 0.9, CR_{min} = 0.1, scaling factor f = 0.5, $\delta = 10^{-4}$. In the calculation, each test functions independently 100 under the same conditions. Each run is initiated using a different set of initial population. The best result (denoted best)of 100 experiments, the average of the optimal value (denoted mean), the worst result (denoted worst), percentage of the successful runs to total runs (denoted ps), the average running time(denoted time) and average number of function evaluations of successful runs (denoted ave) are as follows:

Table 1 demonstrates that the new algorithm (MIPDE) for the 14 test functions has reached the optimal solution basically and the function g 10 (maximization problem) and the function g 07 (minimization problem) are better than the optimal value. All functions ps have reached more than 80% and the ave is relatively small.

New algorithm (MIPDE) is compared with all the three algorithms (MI-LXPM) (Deep *et al*., 2009) RST2ANU (Mohan and Nguyen, 1999) and AXNUM (Li and Gen, 1996) algorithms) in ps ave and time. A run is considered as a success if the achieved value of the objective function is within 1% of the known optimal value (in case the optimal value of the objective is zero, a run is considered as a success if the achieved absolute value of the objective function is less than 0.01). For each problem, the percentage of success (obtained as the ratio of the number of successful runs to total number of runs), the average number of function evaluations in the case of successful runs and the average time in seconds used by the algorithm in achieving the optimal solution in the case of the successful runs are also listed.

It can be seen from Table 2 that each test function MIPDE algorithm's success rate is better than MI-LXPM algorithm and in addition the function 04, 08, 12 and g 14, the percentage of success of all other functions are up to 100%. Based on the average number of function evaluations of successful runs (aveage), functions 01, 02, 08, 09, 10, 11 and g 14 are slightly worse than those of other test functions, the average numbers of function evaluations are better than MI-LXPM algorithm. Based on the average running time (t) terms, functions 01, 05, 06 and g 09, other test functions are better than the average running time of Algorithm MI-LXPM.

It can be seen from Table 3 that the percentage of the success of MIPDE algorithm is better than RST2ANU algorithm for each test function. The percentage of the success of RST2ANU algorithm for functions 03 and g 04 is only 4 and 2%. The function g 07 does not find the optimal solution and for MIPDE algorithm for function 03 and g 04 the percentage of the success can reach 100 and 97% and the percentage of the success on the function of g 07 can reach 100%. From the average number of function evaluations of successful runs' (aveage) point

Table 1: The numerical results

Function/optimal(g)	Best	Mean	Worst	ps	Time	ave
01/2	2	2	2	100	0.05328	9493
02/2.124	2.1244	2.1245	2.1246	100	0.02228	1683
03/1.07654	1.0765	1.0766	1.0771	100	0.15863	9828
04/-6961.81381	-6961.81381	-6950.66668	-5847.40963	97	0.21013	10644
05/-68	-68	-68	-68	100	0.00863	565
06/-6	-6	-6	-6	100	0.00693	36
07/99.245209	99.239635	99.239635	99.239635	100	0.01720	1265
08/3.557463	3.557466	3.560038	3.714353	95	0.30788	13310
09/32217.4	32217.42778	32217.42778	32217.42778	100	0.00801	451
10/0.94347	0.953197	0.952562	0.93977	100	0.03871	1375
11/8	8	8	8	100	0.00425	204
12/14	14	14.24	20	84	0.09780	6293
13/-42.632	-42.6321	-42.6321	-42.6321	100	0.00130	27
14/807	807	807.03	808	97	0.07137	4195

Table 2: Results obtained by using MIPDE and MI-LXPM algorithms

	MI-LXPM			MIPDE		
Problem	ps	ave	t	ps	ave	t
01	84	172	0.03489	100	9493	0.05328
02	85	64	0.05940	100	1683	0.02228
03	43	18608	0.38344	100	9828	0.15863
04	95	10933	0.64642	97	10644	0.21013
05	100	671	0.00234	100	565	0.00863
06	100	84	0.00015	100	36	0.00693
07	59	7447	0.64459	100	1265	0.01720
08	41	3571	0.82012	95	13310	0.30788
09	100	100	0.00032	100	451	0.00801
10	93	258	0.04908	100	1375	0.03871
11	100	171	0.00630	100	204	0.00425
12	71	299979	3.27762	84	6293	0.09780
13	99	77	0.00598	100	27	0.00130
14	92	2437	0.39190	97	4195	0.07137

Table 3: Results obtained by using MIPDE and RST2ANU algorithms

	RST2ANU			MIPDE		
Problem	ps	ave	t	ps	ave	t
01	47	173	0.00229	100	9493	0.05328
02	57	657	0.05211	100	1683	0.02228
03	04	221129	0.19340	100	9828	0.15863
04	02	1489713	172.31000	97	10644	0.21013
05	75	2673	0.00760	100	565	0.00863
06	100	108	0.00512	100	36	0.00693
07	00	-	-	100	1265	0.01720
08	15	180859	4.34473	95	13310	0.30788
09	100	189	0.01030	100	451	0.00801
10	100	545	0.01924	100	1375	0.03871
11	100	2500	0.01095	100	204	0.00425
12	29	6445	0.02431	84	6293	0.09780
13	100	35	0.00343	100	27	0.00130
14	19	3337	0.02821	97	4195	0.07137

of view, functions 01, 02, 09, 10 and g 14 are slightly worse than those of other test functions and the average number of function evaluations are better than RST2ANU algorithm. In the average run Time (t) terms, functions 01, 05, 06, 10, 12 and g 14 are slightly worse, those of other test functions are better than the average running time RST2ANU algorithm.

It can be seen from Table 4 that for each test function the percentage of success of MIPDE algorithm is better than AXNUM algorithm. For AXNUM algorithm functions 08 and g 14 the percentage of success are only 3 and 9%, while the percentage of success of these two functions can reach 95 and 97% for MIPDE algorithm. In the average number of function evaluations of successful runs' (aveage) point of view, functions 01, 02, 08 and g 10 are slightly worse than those of other test functions, the average number of function evaluations are better than AXNUM algorithm. In an average running time (t) terms,

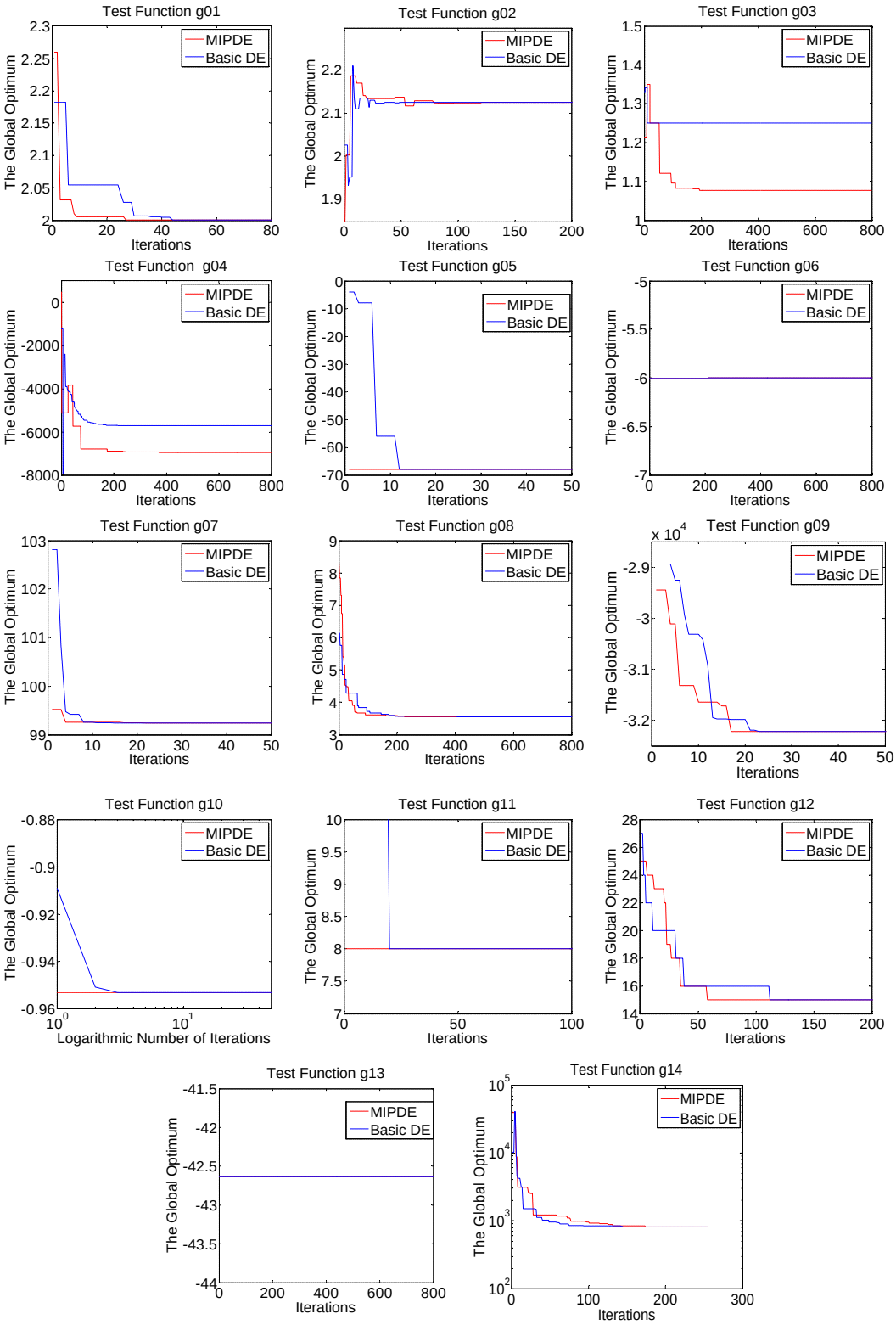

Fig. 1: Explained effectiveness of MIPDE

Table 4: Results obtained by using MIPDE and AXNUM algorithms

	AXNUM			MIPDE		
Problem	ps	ave	t	ps	ave	t
01	86	1728	0.04250	100	9493	0.05328
02	67	82	0.09825	100	1683	0.02228
03	35	65303	0.38677	100	9828	0.15863
04	82	45228	0.22643	97	10644	0.21013
05	95	13820	0.06245	100	565	0.00863
06	100	432	0.00188	100	36	0.00693
07	45	16077	0.64304	100	1265	0.01720
08	03	1950	1.39033	95	13310	0.30788
09	100	4946	0.01691	100	451	0.00801
10	33	700	2.04736	100	1375	0.03871
11	97	863	0.03319	100	204	0.00425
12	19	380115	3.88332	84	6293	0.09780
13	91	456	0.05253	100	27	0.00130
14	09	267177	0.01749	97	4195	0.07137

functions 01, 06 and g 14, other test functions are better than the average running time AXNUM algorithm. The proposed new algorithm has strong searching capability, and its stability is very good.

In order to explain the improved differential evolution algorithm (MIPDE) the effectiveness more directly, the following constraint handling mechanism in the same circumstances draws the basic differential evolution algorithm and our modified differential evolution algorithm for graphical comparison.

From the Fig. 1, we can see that in addition to the function 02 and g 14, the other function when using MIPDE is faster than basic evolutionary differential evolution algorithm; for functions 03 and g 04, the basic differential evolution algorithm is easily trapped into local optimum, and functions can jump out of local optima with MIPDE to find the global minimum, indicating that the proposed new algorithm has strong searching capability and its stability is very good.

Therefore, the experimental results show that the new algorithm can guide the population into the feasible region, quickly search the global optimum and effectively deal with mixed integer programming problems and has good stability and practicality, high success rate and calculated short time.

ACKNOWLEDGMENT

The stduy is supported by the National Natural Science Foundation of China under Grant (60962006). the mutation was modified. The current global optimal solution is added to it. The adaptive adjustment of the position of the current particle is to make the particle swarm for the better direction of the evolution, using the rules of the feasible basis and dynamic constraint handling technology to improve the selectivity of particles' optimization.

REFERENCES

Bergamini, M.L., I. Grossmann and N. Scenna, 2008. An improved piecewise outer-approximation algorithm for the global optimization of MINLP models involving concave and bilinear terms. Comput. Chem. Eng., 32: 477-493.

Chiou, J.P. and F.S. Wang, 1998. A hybrid method of differential evolution with application to optimal control problems of a bioprocess system. Proceedings of the IEEE World Congress on Computational Intelligence, May 4-9, Anchorage, AK., pp: 627-631.

Costa, L. and P. Oliveira, 2001. Evolutionary algorithms approach to the solution of mixed integer non-linear programming problems. Comput. Chem. Eng., 25: 257-266.

Deb. K., 2000. An efficient constraint handling method for genetic algorithms. Comput. Methods Applied Mech. Eng., 186: 311-338.

Deep, K., K.P. Singh, M.L. Kansal and C. Mohan, 2009. A real coded genetic algorithm for solving integer and mixed integer optimization problems. Applied Math. Comput., 212: 505-518.

Deng, Z.X., 2008. A new differential evolution algorithm. Comput. Eng. Appl., 44: 40-42.

Li, H., Y. Jiao and L. Zhang, 2008. Solving mixed integer programming hybrid evolutionary algorithm. Control Decis., 23: 1098-110.

Li, Y.X. and M. Gen, 1996. Nonlinear mixed integer programming problems using genetic algorithm and penalty function. Proceeding of the IEEE International Conference on Systems, Man and Cybernatics, Oct. 14-17, Beijing, pp: 2677-2682.

Lin, Y.C., K.S. Hwang and F.S. Wang, 2000. Plant scheduling and planning using mixed-integer hybrid differential evolution with multiplier updating. Proceedings of the Congress on Evolutionary Computation, July 16-19, La Jolla, CA., USA., pp: 593-600.

Lin, Y.C., K.S. Hwang and F.S. Wang, 2004. A mixed-coding scheme of evolutionary algorithms to solve mixed-integer nonlinear programming problems. Comput. Math. Appl., 47: 1295-1307.

Lu, H. and W. Chen, 2006. Dynamic-objective particle swarm optimization for constrained optimization problems. J. Comb. Optim., 12: 409-419.

Lu, H. and W. Chen, 2008. Self-adaptive velocity particle swarm optimization for solving constrained optimization problems. J. Global Optim., 41: 427-445.

Mohan, C. and H.T. Nguyen, 1999. A controlled random search technique incorporating the simulated annealing concept for solving integer and mixed integer global optimization problems. Comput. Optim. Appl., 14: 103-132.

Nowak, I. and S. Vigerske, 2008. LaGO: A (heuristic) branch and cut algorithm for nonconvex MINLPs. Cent. Eur. J. Oper. Res., 16: 127-138.

Storn, R. and K. Price, 1995. Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report. International Computer Science Institute, Berkley, TR-95-012, California.

Storn, R. and K. Price, 1997. Differential evolution-A simple and efficient heuristic for global optimization over continuous spaces. J. Global Optimization, 11: 341-359.

Vanderbeck, F. and M.W.P. Savelsbergh, 2006. A generic view of Dantzig-Wolfe decomposition in mixed integer programming. Oper. Res. Lett., 34: 296-306.

Yokota, T., M. Gen and Y.X. Li, 1996. Genetic algorithm for non-linear mixed integer programming problems and its applications. Comput. Ind. Eng., 30: 905-917.