Structure-Encoding Differential Evolution for Integer Programming

Changshou Deng
Institute of Computer Network System, Hefei University of Technology, Hefei City, China
Email: csdeng@jju.edu.cn

Changyong Liang
School of Management, Hefei University of Technology, Hefei City, China
Email: cyliang@163.com

Bingyan Zhao
School of Business, Jiujiang University, Jiujiang City,Chian
Email: petramm@163.com

Yanlin Yang and Anyuan Deng
School of Information Science and Technology, Jiujiang University, Jiujiang City, China
Email: {yangyl, dengay}@jju.edu.cn

***Abstract*—Differential Evolution is a competive method for continuous number optimization problems. A novel Structure-Encoding Differential Evolution (SEDE) algorithm was proposed for optimization problems with integer-parameter representation. In the SEDE Algorithm, each decision variable of every individual consists of two domains. One domain is float-encoding which is confined in a narrow range [0, 1]. The other domain is integer-encoding which is used to represent the problem space. A new operator, boundary-handling operator, was used to ensure each result generated by the mutation operator falling into the range [0, 1]. In addition, a new mapping operator was constructed to generate integer number from the real domain. The global convergence property of the SEDE was analyzed. The simulation results of several Benchmarks of integer programming show it is effective and efficient. Structure-encoding Differential Evolution algorithm is a new effective way for handling the integer programming problems.**

***Index Terms*—Integer Programming, Structure-encoding Differential Evolution, boundary-handling operator, mapping operator**

I. INTRODUCTION

Differential Evolution (DE), proposed by Storn and Price[1], is a simple yet powerful algorithm for global optimization over continuous spaces, which use the greedy criterion to make decision. Recently, the DE algorithm has become quite popular in the machine intelligence and cybernetics. It has been successfully been applied to diverse fields of science and engineering, such as mechanical engineering design [2], signal processing [3] and pattern recognition [4]. It has been proved to perform better than the Genetic Algorithm (GA) or the Particle Swarm Optimization (PSO) by numerical benchmarks experiments [5]. On the other hand, the DE structure has some limitations in the search logic, since it contains too narrow a set of exploration moves [6].There are many modifications to the original algorithm. Fan and Lampinen(2002) [7]proposed a trigonometric DE (TDE). In the TDE, a new mutation called trigonometric mutation replaced the traditional mutation with certain probability. The trigonometric mutation has the role of promoting the generation of the offspring along optimal directions. In this sense this special mutation can be considered as a speical local refining operation. Noman and Iba (2008) [8] proposed a memtic approach to accelerate the speed of DE. The main idea is that a proper balance of the exploration abilities of DE and the exploitation abilities of a local search can enhance the performance of DE.

Although DE has been successful in numerical optimization, only a few works concern its usage for discrete optimization problems [9].

However, a remarkably wide variety of problems can be represented by discrete optimization models in effect. For instance, many applications in Operational Research such as goods distribution, production scheduling, and machine sequencing are encountered. In most of them, integer programming problems are met [10]. And recent years, the neural networks with integer weights have been promising since this kind of neural networks is better suited for hardware implementation.

The Integer Programming problem can be presented as

$$\min\ f(x), \quad x \in S \subseteq Z^n, \tag{1}$$

where Z is the set of integers, and S is a not necessarily bounded set.

Evolutionary algorithms applied on real search space can be used on such problems and determine the optimum solution by rounding off the real optimum values to

Corresponding author: Changshou Deng (csdeng@jju.edu.cn)

doi:10.4304/jsw.6.1.140-147

the nearest integer. However, the rounding might result in a value of the objective function that is far removed from the optimum [12].

Despite the simplicity and successful application in many engineering fields and many improvements, its application on the solution of integer optimization problems with integer decision variables is still unusual. One of the possible reasons for this lack is that DE cannot keep the closure when the original DE operators are used in integer domain directly, for the operators designed in the original DE are designed only for continuous domain. A new structure-encoding DE was proposed in this paper to apply the DE algorithm in handling integer programming problem while keeping the simplicity and high efficiency in original DE algorithm.

The rest of the paper is organized as follows. Section II gives a brief introduction of original DE. SEDE with the capability to solving the integer programming problems is given in section III. Several benchmarks of integer programming problems are used to evaluate this structure-encoding DE in section IV. Section V concludes this paper.

II. DIFFERENTIAL EVOLUTION ALGORITHM

Like other Evolutionary Algorithms (EAS), DE is a population-based stochastic optimizer that starts to explore the search space by sampling at multiple, randomly chosen initial points.

It is a kind of float point encoding evolutionary optimization algorithm. AT present, there have been several variants of DE [1]. One of the most promising schemes, DE/rand/1/bin (DE/best/1/bin) scheme of Storn & Price, is presented in great detail. The pseudo code of DE is given as follows.

Initial Population Generation
REPEAT
Mutation Operator
Crossover Operator
Selection Operator
UNTIL (termination criteria are met)

A. Generation of Initial Population

The DE Algorithm starts with the initializing target population $X = (x_{ij})_{m\times n}$ with the population size m and the dimension n, which is generated by the following way.

$$x_{ij}(0) = x_j^l + rand(0,1)(x_j^u - x_j^l) \tag{2}$$

where $i = 1,2,\cdots,m$, $j = 1,2,\cdots,n$, x_j^u denotes the upper constraints, and x_j^l denotes the lower constraints.

B. Mutation Operator

For the scheme DE/rand/1/bin, each target vector $x_i(i = 1,2,\cdots,m)$, a mutant vector is produced by formula (3a).

$$h_i(t+1) = x_{r1} + F(x_{r2} - x_{r3}) \tag{3a}$$

where $i, r_1, r_2 \in \{1,2,\ldots,m\}$ are randomly chosen and must be different from each other. And F is the scaling factor which has an effect on the difference between the individual x_{r1} and x_{r2}.

Basically, scheme DE/best/1/bin works the same way as DE/rand/1/bin except that it generates the vector $h_i(t+1)$ according to formula (3b):

$$h_i(t+1) = x_{best} + F(x_{r2} - x_{r3}) \tag{3b}$$

where x_{best} is the best solution in the current generation.

C. Crossover Operator

DE employs the crossover operator to add the diversity of the population. The approach is given below.

$$u_i(t+1) = \begin{cases} h_i(t+1), & if \quad rand \le CR \; or \\ & j = rand(i) \\ x_i(t), & otherwise \end{cases} \tag{4}$$

where $i = 1,2,\cdots,m$, $j = 1,2,\cdots,n$, $CR \in [0,1]$ is crossover constant and $rand(i) \in (1,2,\cdots n)$ is the randomly selected index. In other words, the trial individual is made up with some of some components of the mutant individual, or at least one of the parameters randomly selected, and some of other parameters of the target individual.

D. Selection Operator

To decide whether the trial individual $u_i(t+1)$ should be a member of the next generation, it is compared to the corresponding $x_i(t)$. The selection is based on the survival of the fitness among the trial individual and the corresponding one such that:

$$x_i(t+1) = \begin{cases} u_i(t+1), & if \quad f(u_i(t+1)) < f(x_i(t)) \\ x_i(t), otherwise \end{cases} \tag{5}$$

III. STRUCTURE-ENCODING DE

The mutation operator by formula (3a) and formula (3b) show that the original DE cannot keep the closure when it is directly applied in solving integer programming problems. For the mutation operator in DE is designed only for the problems in the continuous domain. In order to overcome this shortcoming of the DE, a structure code was proposed. Each decision variable has two field, one is float-encoding which is confined in a narrow interval [0,1]. The other field is integer-encoding which is used to represent the decision variable. A new DE using structure code is called structure-encoding DE. The SEDE is described as fellows.

A. Boundary constraints handling operator

In SEDE, even the results of mutation and crossover operator on the float field may violate the boundary constraints. That is to say the result may falls outside the range $[0,1]$ after being mutated. How to keep the results of mutation and crossover closure should be considered. A new boundary constraint handling operator was defined to replace mutation result which violates the boundary constraints. The boundary constraints handling operator can be defined as (6).

$$u_i(t+1)=\begin{cases}u_i(t+1), if\ u_i(t+1)\in[0,1]\\ rand(0,1), \quad otherwise\end{cases} \tag{6}$$

where $rand(0,1)$ is a new one randomly generated by the SEDE.

B. Code mapping operator

A new mapping operator was defined to connect the float field and the integer field. Firstly, the range $[0,1]$ was partitioned into n equal sub ranges, such as $[0,1/n),[1/n,2/n),\ldots[(n-1)/n,1]$. Then a one to one mapping can be constructed between the sub ranges and the integers. Hence, it is very easy to determine the only integer number by the inverse image x which is in the range $[0,1]$. The code mapping operator can be defined as a mapping function f , which can be defined as (7).

$$f(x)=\begin{cases}1 & ,x\in[0,1/n)\\ \ldots & ,\ldots\\ n & ,x\in[(n-1)/n,1]\end{cases} \tag{7}$$

The Matlab source code of the code mapping operator for the following function F1 with initial interval $[-100,100]$ is presented in Fig. 1.

```
function intery=submap(x01)
%
M=100;
y=-M:M;
M2Mplus1=2*M+1;
M2M=1/M2Mplus1;
% y=-10:10;
x=0:M2M:1;
if x01<M2M
   intery=-M;
   else

   for i=1:M2Mplus1
      if x01>x(i) & x01<=x(i+1)
      intery=y(i);
      break;
      end
   end
end
```

Figure 1 Source code of the code mapping operator

C. Structure-encoding population

In the SEDE, each individual includes two equal vectors. The first vector is a float-pointed vector X , the other one is a integer vector I . The sub vector X is the vector used in original DE, and the sub integer vector I is the image of the vector X mapped by Code mapping operator. The Structure-encoding of each individual can be denoted by formula (8).

$$Y=(X,I) \tag{8}$$

where $I=f(X)$. And Y which is made up of a float-point vector and a integer vector is called Structure-encoding population.

An example of 10 individuals of the following example F_1, with D=5, initial interval $[-100,100]$, is shown in table I. For the limited space, only the first three decision variables are given.

D. Alternative Mutation

In order to make good use of the exploration advantage of scheme DE/rand/1/bin, and the exploitation advantage of the scheme DE/best/1/bin. A new mutation mechanism called alternative mutation is adopted in the SEDE.

Given probability λ , the alternative mutation is as formula (9)

$$h_i(t+1)=\begin{cases}x_{r1}+F(x_{r2}-x_{r3}), & if\ (\lambda<rand)\\ x_{best}+F(x_{r2}-x_{r3}), & otherwise\end{cases} \tag{9}$$

where λ is controlled by formula (10).

$$\lambda=2-e^{(iter/iter\max*\log(2))} \tag{10}$$

where *iter* is the current generation, and *iter* max is the maximum iterations predefined. The probabilities for the two mutation mechanism vary with generation are given in Fig. 2.

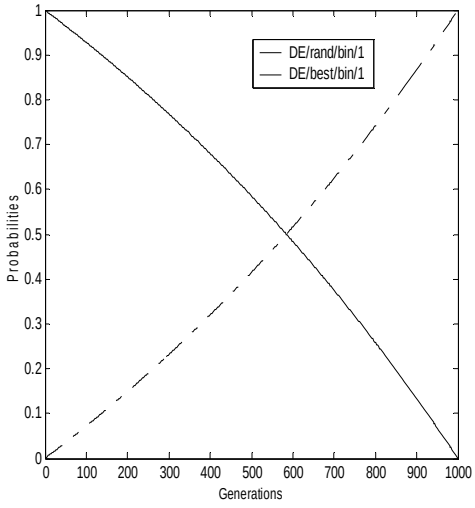

Figure 2 Probabilities vary with Generations

It is very easy to understand that the probability for DE/rand/bin/1 decreases when the generation grows. And the probability for the mechanism DE/best/bin/1 increases when the generation grows. In this way, at the beginning of the evolution, exploration the space has the higher probability. At the last stage of evolution, exploitation the space has the higher probability. With this new mutation scheme, the performance of the SEDE was improved by the new scheme of balancing the exploration and exploitation.

E. Steps of SEDE

Steps of the SEDE are as fellows.

Step1: Generation of intial population. In the population, one field is float-encoding which is confined in [0,1] and the other field is integer-encoding which is generated by the code mapping operator.

Step2*:* Evalutation of the fitness function.

Step3: If the termination condition is true, then SEDE terminates. Otherwise, go to step4.

Step4: Alternative Mutation

Step5: Crossover operator

Step6: code mapping operator

Step7: selection operator

Step8: go to step3

The flowchart of the SEDE is illustrated as Fig. 3.

F. Convergence analysis of SEDE

In this section, a new way proposed in [12] was used to analyze the convergence of the proposed SEDE.

Let $X_G = (x_{1,G}, x_{2,G}, \ldots, x_{m,G}), (G = 1, \ldots, G_{\max})$ be the random population of size m at step $G \geq 0$, and $F_G = \min\{f(x_{i,G}) : i = 1, \ldots, m\}$ be the best fitness value within the population at step $G \geq 0$, When the random F_G variable contains the value of the global optimum, f^*, it is confirmed that one individual in the population representing the global solution of the minimization problem. Ideally, this condition should be met after a finite number of iterations with probability one regardless the initialization population of the SEDE.

Property 1. In the SEDE, it is well known that the best solutions found in the current generation will be carried over to the next generation. This property ensures that the optimum will be located in finite time and never be lost once it is found. Thus, the property above shows that the random sequence $(F_G, G \geq 0)$ converges to the optimum, f^*.

TABLE I.
EXAMPLE OF STRUCTURE-ENCODING POPULATION

Y_1		Y_2		Y_3	
X_1	I_1	X_2	I_2	X_3	I_3
0.4091	-18	0.0754	-85	0.4750	-5
0.5355	7	0.7961	60	0.4469	-11
0.2461	-51	0.8292	66	0.6554	31
0.6577	32	0.5454	9	0.1352	-73
0.9118	83	0.8955	79	0.7100	42
0.0747	-85	0.3384	-32	0.9463	90
0.8341	67	0.5627	13	0.2392	-52
0.0390	-93	0.6884	38	0.6743	35
0.6096	22	0.5271	5	0.8653	73
0.5573	12	0.8947	79	0.2171	-57

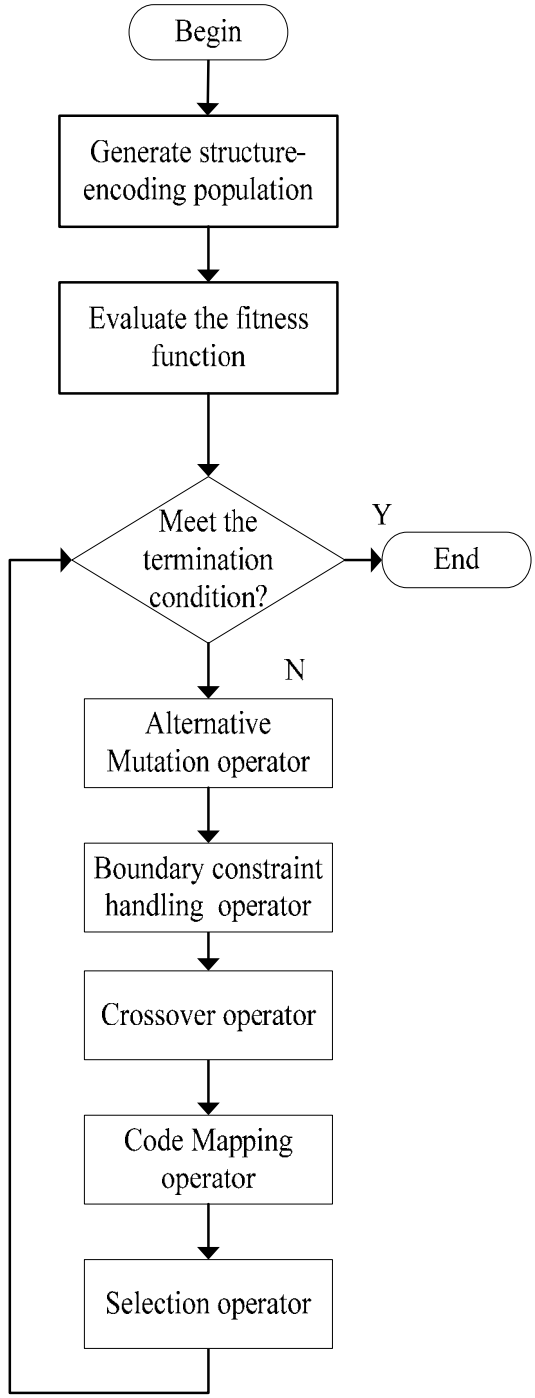

Figure 3. Flowchart of the structure-encoding DE

Let $(x_1, x_2, \ldots, x_m) \in X^n$ denote the current population of parents. An offspring is generated as fellows. Firstly, m numbers of parents are chosen to serve as elder individuals to generate a candidate individual. This operator is presented as follows.

$$eld: \quad X^m \to X^j \quad where \quad 3 \le j \le m \tag{11}$$

These individuals are then used to generate the candidate individual by the alternative mutation operator, which can be abstracted as following.

$$am: \quad X^j \to X. \tag{12}$$

Secondly, crossover with the base individuals, a trial individual is produced by the following process

$$cor: \quad X \to X. \tag{13}$$

Finally, the selection operator will determine which ones will remain as the new parents in the next generation. The selection operator is also abstracted as

$$sel: X \to X. \tag{14}$$

In a word, a single generation of the structure encoding DE can be presented as follows.

$$\begin{aligned} &\forall i \in \{1, \ldots, m\}: \\ &v_i = cor(am(eld(x_1, \ldots, x_m))) \end{aligned} \tag{15}$$

$$\forall i \in \{1, \ldots, m\}: x_i = \{sel(x_i, v_i)\} \tag{16}$$

After the abstract description of the SEDE, some assumptions about the above operators are defined.

Assumption 1. Each individual may be chosen to be a parent individual and can be changed to an arbitrary other individual by a finite number of successive alternative mutation, i.e., for each $x \in X$ there exists a finite path such that

$$\Pr\{x_i = am(x_j)\} = 1, i \ne j \text{ and } i, j \in [1, m] \tag{17}$$

Assumption 2. Each candidate individual is changed by the crossover operator with the minimum probability $p_{cr} \succ 0$.

$$\begin{aligned} &\forall i \in \{1, \ldots, m\}: \\ &\Pr = \{v_i = cor(am(eld(x_1, \ldots, x_m)))\} \ge p_{cr} \succ 0 \end{aligned} \tag{18}$$

Assumption 3. Each trial individual competes for the next generation with a minimum probability 0.5.

$$\forall i \in \{1, \ldots, m\}: \Pr\{sel(x_i, v_i)\} = 0.5 \tag{19}$$

Theorem 1. If the Assumption 1-3 are valid, then the SEDE visits the global optimum after a finite number of generations with probability one regardless of the initialization.

Proof. Let random variable $T = \{G \ge 0 : F_G = f^*\}$ denote the first hitting time of the global solution. An evolutionary algorithm is said to visit the global optimum in finite time with probability one if $\Pr\{T \prec \infty\} = 1$ regardless of the initialization.

Let $X^* = \{x \in X, f(x) = f^*\}$ be the set of globally optimal solutions. According to Assumption 1, there exists a finite path from an arbitrary $x \notin X^*$ to some $x^* \in X^*$ that can be traversed by successive alternative mutation. Let Lx be the length of the path between $x \notin X^*$ and the set $x^* \in X^*$.

Considered an arbitrary individual x of some population is chosen as the parent individual. Assumption 1 ensures that this parent passes the alternative mutation process with every change with probability one. The probability that the candidate individual transits to the next point of the path towards $x^* \in X^*$ by crossover is ensured to be at least $p_{cr} \succ 0$ by assumption2.

With Assumption 3, the offspring will remain in the next generation with probability 0.5. Thus, the probability that parent $x \notin X^*$ transits to a parent representing the next point on the path to $x^* \in X^*$ is at least $0.5 p_{cr} \succ 0$. Consequently the probability that a global optimal solution has not been found is $(1 - 0.5 p_{cr})$.

After G_w generations, the SEDE has not found the optimal solution is at most $(1 - 0.5 p_{cr})^{G_w}$.

For the SEDE,

$$\Pr\{T \prec \infty\} = \lim_{G_w \to \infty} (1 - (1 - 0.5 p_{cr})^{G_w}) = 1 \tag{20}$$

Thus, a global optimum will be found for the first time after a finite number of iterations with probability one.

IV. Experiments

In this section, results regarding the performance of the SEDE on the class of problems known as integer programming are reported. Seven integer programming problems used in literature [11] which was used to verify the performance of PSO were selected to investigate the performance of the SEDE method. And the comparison between the performances of the SEDE with that of PSO is also reported.

A. Test Problems

The test problems are defined immediately below:

Test Problem 1 (Rudolph,1994) [13],

$$F_1(x) = \|x\|_1 = |x_1| + \ldots |x_D| \tag{21}$$

where D is the corresponding dimension. The olution is $x_i^* = 0, i = 1, \ldots, D$ with $F_1(x^*) = 0$.

Test Problem 2 (Rudolph,1994) [13],

$$F_2(x) = x^T x \tag{22}$$

where D is the corresponding dimension. The solution is $x_i^* = 0, i = 1, \ldots, D$ with $F_1(x^*) = 0$.

Test Problem 3 (Glankwahmdee et al., 1979) [14],

$$F_3(x) = -(15\ 27\ 36\ 18\ 12)x + xT\begin{bmatrix} 35 & -20 & -10 & 32 & -10 \\ -20 & 40 & -6 & -31 & 32 \\ -10 & -6 & 11 & -6 & -10 \\ 32 & -31 & -6 & 38 & -20 \\ -10 & 32 & -10 & -20 & 31 \end{bmatrix} x \quad (23)$$

with best known solution $x^* = (0,11,22,16,6)^T$ and $x^* = (0,12,23,17,6)^T$, with $F_3(x) = -737$.

Test Problem 4 (Glankwahmdee et al., 1979) [14],

$$F_4(x) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2, \quad (24)$$

with solution $x^* = (3,2)^T$ and $F_4(x^*) = 0$.

Test Problem 5 (Glankwahmdee et al., 1979) [14],

$$F_5(x) = (9x_1^2 + 2x_2^2 - 11)^2 + (3x_1 + 4x_2^2 - 7)^2, \quad (25)$$

with solution $x^* = (1,1)^T$ and $F_5(x) = 0$.

Test Problem 6 (Glankwahmdee et al., 1979) [14],

$$F_6(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2, \quad (26)$$

with solution $x^* = (1,1)^T$ and $F_6(x) = 0$.

Test Problem 7 (Glankwahmdee et al., 1979) [14],

$$F_7(x) = (x_1 + 10x_2)^2 + 5(x_3 - x_4) + (x_2 - 2x_3)^4 + 10(x_1 = x_4)^4 \quad (27)$$

with solution $x^* = (0,0,0,0)^T$ and $F_7(x) = 0$.

B. Parameters Set

The SEDE parameters used for all experiments were $F = 0.5$ and $CR = 0.3$.

For the test functions F_1 and F_2, several experiments were performed for different dimension. For all experiments the initial was taken uniformly distributed inside $[-100,100]^D$, where D is the corresponding dimension. The population size as well as the maximum number of iterations, for each dimension, for the first two test functions, is given in table II. For fair comparison, the parameters are just same as the PSO [13] used.

For the test functions $F_3 - F_7$, the population size was fixed. The population size, the maximum number of iterations for each of these test functions are exhibited in Table III.

C. Experimental Results

For each dimension and for each function, 100 experiments were performed for the test functions F_1 and F_2. The success rate, the mean number of structure-encoding iterations as well as the mean number of function evaluations for each dimension of F_1 and F_2 are reported in table IV and table V respectively.

For the test functions $F_3 - F_7$, the success rate, the mean number of iterations and the mean number of function evaluations for each of the interval $[-100,100]^D$, $[-50,50]^D$, $[-25,25]^D$, where D is the corresponding dimension of the functions, are reported in Tables VI-VIII.

It can be easily seen from table IV and table V that the SEDE can reach the optimum every time in 100 independent runs for the different dimensions. It takes only nearly one fifth in number of function evaluation compared with the PSO. For the function F_1 with dimension 30 and the function F_2 with dimension 30, the success rate of the SEDE is higher than that of the PSO. For both F_1 and F_2 with dimension 5, the convergence speed of the SEDE is almost as ten times as that of the PSO.

Table VI shows us that for F_3 and F_7 with interval $[-100,100]^D$, the success rates of the SEDE 100%, while the success rate of the PSO for F_3 and F_7 are 80% and 92 respectively. The success rates of the other three functions of the SEDE and the PSO are the same. For each function, the convergence speed of the SEDE is higher than that of the PSO both in mean number of iterations and mean number of function evaluations

Table VII and Table VIII tell us that for function F_3 with interval $[-50,50]^D$ and function F_3 with interval $[-25,25]^D$), the success rate of the SEDE is higher than that of the PSO. For the other functions, the SEDE and the PSO have the same success rate. However, the SEDE has higher speed to reach the optimum for all the functions concerned.

In a word, the experimental results indicate the proposed SEDE is an efficient method and should be considered as a good alternative to handle Integer Programming problems. The behavior of the SEDE seems to be robust even for high dimensional cases, exhibiting very high success rates of even with modest population size.

V. CONCLUSION

DE is a recently developed evolutionary algorithm that has empirically proven to be very robust for global optimization over continuous spaces. A novel SEDE was proposed to enable DE to operate within integer spaces. And an alternative mutation operator was used to balance the exploration and exploitation ability of the SEDE. The SEDE was proven to reach the global optimum with probability one.

The experiments results on several benchmark integer programming problems show that the performance of the SEDE was very favorable. It is robust and has fast convergence speed. Compared with PSO, it is nearly five

times faster than that of PSO in terms of Mean of function evaluation for most of the cases.

The main benefit is that the SEDE can operate in the integer spaces while keeping the advantage of the traditional DE. The method proposed in this paper can also be applied to other float-encoding EA for the Integer Programming problems.

TABLE II
POPULATION SIZE AND MAXIMUM NUMBER OF ITERATIONS FOR DIFFERENT DIMENSION'S VALUES, FOR FUNCTION F_1AND F_2.

D	POPLATION SIZE	Max. It.
5	20	1000
10	50	1000
15	100	1000
20	200	1000
25	250	1000
30	300	2000

TABLE III
DIMENSIONS, POPULATION SIZE AND MAXIMUM NUMBER OF ITERATIONS, FOR FUNCTIONS F_3-F_7

Function	D	POPLATION SIZE	Max. It.
F3	5	50	1000
F4	2	20	1000
F5	2	20	1000
F6	2	20	1000
F7	4	40	1000

TABLE IV
SUCCESS RATE, MEAN NUMBER FO ITERATIONS AND MEAN NUMBER OF FUNCTION EVALUATIONS FOR DIFFERENT DIMENSIONS OF THE FUNCTION F_1.

Function (D)	Method	Suc.Rate	Mean Iter	Mean F.Eval.
F1(5）	PSO	100%	440.72	8834.3
	SEDE	100%	**47.45**	**949**
F1(10)	PSO	100%	453.48	22724.0
	SEDE	100%	**86.86**	**4343**
F1(15)	PSO	100%	459.44	46044.0
	SEDE	100%	**112.5**	**12250**
F1(20)	PSO	100%	465.24	93248.0
	SEDE	100%	**156.93**	**31386**
F1(25)	PSO	100%	683.44	171110
	SEDE	100%	**198.51**	**49628**
F1(30)	PSO	80%	914.64	274692
	SEDE	100%	**235.83**	**47166**

TABLE V
SUCCESS RATE, MEAN NUMBER FO ITERATIONS AND MEAN NUMBER OF FUNCTION EVALUATIONS FOR DIFFERENT DIMENSIONS OF THE FUNCTION F_2.

Function (D)	Method	Suc.Rate	Mean Iter	Mean F.Eval.
F2(5)	PSO	100%	440.92	8838.4
	SEDE	100%	**49.11**	**982.2**
F2(10)	PSO	100%	454.88	22794
	SEDE	100%	**94.74**	**4737**
F2(15)	PSO	100%	462.88	46388
	SEDE	100%	**136.65**	**13665**
F2(20)	PSO	100%	467.08	93616
	SEDE	100%	**174.62**	**34924**
F2(25)	PSO	100%	685.44	171610
	SEDE	100%	**214.64**	**42928**
F2(30)	PSO	84%	914.4	274692
	SEDE	100%	**256.9**	**51380**

TABLE VI
SUCCESS RATES, MEAN NUMBER OF ITERATIONS AND MEAN NUMBER OF FUNCTION EVALUATIONS FOR F_3-F_7 WITH INITIAL INTERVAL $[-100,100]^D$.

Function (D)	Method	Suc.Rate	Mean Iter	Mean F.Eval.
F3	PSO	80%	499.4	75060
	SEDE	100%	**517.29**	**25865**
F4	PSO	100%	402.32	8066.4
	SEDE	**100%**	**97.71**	**1954.2**
F5	PSO	100%	415.08	8321.6
	SEDE	100%	**13.97**	**279.4**
F6	PSO	100%	418.4	8388
	SEDE	100%	**199**	**3980**
F7	PSO	92%	460.92	18476.8
	SEDE	**100%**	**137.65**	**2753**

Acknowledgment

This work was supported in part by Natural Science Foundation of China with grant no.70771037 and the Science and Technology Project of Jiangxi Province, China with grant no.GJJ10616.

TABLE VII
SUCCESS RATES,MEAN NUMBER OF ITERATIONS AND MEAN NUMBER OF FUNCTION EVALUATIONS FOR F_3-F_7 WITH INITIAL INTERVAL $[-50,50]^D$.

Function (D)	Method	Suc.Rate	Mean Iter	Mean F.Eval.
F3	PSO	92%	447.72	67308
	SEDE	100%	**415.7**	**20785**
F4	PSO	100%	324.48	6589.6
	SEDE	100%	**84.03**	**1680.6**
F5	PSO	100%	357.12	7162.4
	SEDE	100%	**10.22**	**204.4**
F6	PSO	100%	324.64	6872.8
	SEDE	100%	**47.06**	**941.2**
F7	PSO	100%	440.8	17672
	SEDE	100%	**103.2**	**2060.4**

TABLE VIII
SUCCESS RATES,MEAN NUMBER OF ITERATIONS AND MEAN NUMBER OF FUNCTION EVALUATIONS FOR F_3-F_7 WITH INITIAL INTERVAL $[-25,25]^D$.

Function (D)	Method	Suc.Rate	Mean Iter	Mean F.Eval.
F3	PSO	96%	420.28	63192
	SEDE	100%	**308.64**	**15432**
F4	PSO	100%	402.32	8066.4
	SEDE	100%	**20.79**	**415.8**
F5	PSO	100%	228.28	4585.6
	SEDE	100%	**7.46**	**149.2**
F6	PSO	100%	332.08	6661.6
	SEDE	100%	**18.72**	**374.4**
F7	PSO	100%	429.32	17212.8
	SEDE	100%	**71.26**	**1425.2**

REFERENCES

[1] R. Storn and K. Price, "Differential Evolution-A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *Journal Global Optimization*, Vol. 11, pp.241-354, 1997.

[2] Rogalsky, Derksen, and Kocabiyik, " Differential evolution in aerodynamic optimization", Proc. of 46th Annual Conf. of Canadian Aeronautics and Space Institute, pp. 29-36,1999.

[3] S. Das and A. Konar, "Design of tow dimensional IIR filters with modern search heuristics: a comparative study", International Journal of Computational Intelligence and Applications, World Scientific Press, Vol.6 No.3, pp.176-185.,2006.

[4] S. Das, A. Abraham and A. Konar, "Adaptive clustering using improved differential evolution algorithm", IEEE Transaction on Systems, Man and Cybernetics-Part A, IEEE Press, USA, vol.38, issue 1:pp. 218-237, 2008.

[5] J. Versterstrom, R. Thomsen " A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithm on numerical benchmark problems" Evolutionary Computation, CEC2004.Portland OR: IEEE press, 2: 1980-1987,2004.

[6] N. Ferrante, V. Tirronen, "Recent advances in differential evoltuion," *Artif Intell Rev.*, Vol. 10, pp.:1-46, October 2009.

[7] H.Y. Fan, J.Lampinen, "A trigonometric mutation approach to differential evolution," Evolutionary methods for design, optimization and control. CIMNE, Barcelona, pp 65–70,2002.

[8] N.Noman , H.Iba, "Accelerating differential evolution using an adaptive local search,". *IEEE Trans Evol Comput* Vol. No.1:107–125, 2008.

[9] T. Gong, L. T. Andrew, "Differential Evolution for Binary Encoding," *Soft Computing in Industrial Applications*, ASC 39, pp.251-262, 2007.

[10] G.L.Nemhauser , Rinooy Kan AHG and Todd MJ, *Handbooks in OR & MS, Vol.1: Optimization.* Elsevier, 1989.

[11] SS Rao, *Engineering optimization-theory and practice*, Wiley, 1996.

[12] B. Subudhi, J. Debashisha, "An improved differential evolution trained neural network scheme for nonlinear system identification", *International Journal of Automation and Computing*, Vol.6, No.2, pp. 137-144, 2009.

[13] Parsopoulos K.E. ,Vrahatis M.N. "Recent approaches to global optimization problems through Particle Swarm optimization", *Natural Computing*, 2002,1:235-306.

[14] G. Rudolph, "An evolutionary algorithm for integer programming," In: Davidor Y, Schwefel H-P, Männer R (eds),. Parallel Problem Solving from Nature, pp. 139–148, 1994

[15] A.Glankwahmdee , J.S. Liebman JS and Hogg GL , " Unconstrained discrete nonlinear programming," *Engineering Optimization* , Vol.4: pp. 95–107, 1979

Changshou Deng, born in Anhui Province, China. InAugust, 1972. In 1995, 2001 he got the bachelor degree (Computer Engineering) and master degree (Computer Engineering) respectively from Yanshan University, China. He earned the doctor degree from Tianjin University, China, majoring in Management Science and Engineering in 2007. He is now in the postdoctoral program in Hefei University of Technology.

He is an associate professor in Jiujiang University, China. He has written more than ten articles about DE algorithm. He is a member of CCF(China Computer Foundation),China. His main research field is about Evolutionary Algorithm and Data Mining.