ISSN: 2393-8528

IJICSE
Open Access

Contents lists available at www.ijicse.in

International Journal of Innovative Computer Science & Engineering

Volume 3 Issue 5; September-October-2016; Page No. 17-24

New genetic infused algorithm for problems of optimization

Prof.(Dr.)Vijay Kumar, Director, Principal, RDJ Group of Institutions, Basoli, Punjab, India

Dr.Rakesh Kumar, Assistant Professor, PIT, Bhikhiwind, Punjab, India

ARTICLE INFO

Received: 03 September 2016
Accepted: 26 September 2016

Corresponding Author:

Prof.(Dr.)Vijay Kumar

ABSTRACT

In the paper a novel hybrid genetic differential evolution algorithm utilized to solve constrained optimization problems. The suggested algorithm is called hybrid genetic differential evolution algorithm for solving constrained optimization problems. The objective of the suggested algorithm has to mend the global search aptitude of DE algorithm by merging genetic linear crossover with a DE algorithm to discover more solutions in search space and to revoke tricking in local minima. To validate the general performance of the differential evolution algorithm, it compared with 4 evolutionary based algorithms on 8 benchmark functions. The experimental results show that optimal algorithm

Keywords- optimization , Genetic algorithms, mutation, generation

Introduction

Evolutionary algorithms widely used to solve many unconstrained optimization problems. EAs are unconstrained search algorithms and lake a technique to handle the constraints in the constrained optimization problems (COPs). There are different techniques to handle constraints in EAs, these techniques classified by Michalewicz as follows. Methods based on penalty functions, methods based on the rejection of infeasible solutions, methods based on repair algorithms, methods based on specialized operators and methods based on behavioral memory.

Differential evolutionary algorithm is one of the most widely used evolutionary algorithms (EAs) introduced by Stron and Price. Because of the success of DE in solving unconstrained optimization problems, it attracts many researchers to apply it with their works to solve constrained optimization problems (COPs). In this paper, we proposed a new hybrid algorithm in order to solve constrained optimization problems. The proposed algorithm is called hybrid genetic differential evolution algorithm for solving constrained optimization problems (HGDESCOP). The HGDESCOP algorithm starts with an initial population consists of NP individuals, the initial population is evaluated using objective function. At each generations, the new offspring created by applying the DE mutation. In order to increase the global search behaviour of the proposed algorithm and explore wide area of the search space, a genetic algorithm linear crossover operator is applied. In the last stage of the algorithm, the greedy selection is applied in order to accept or reject the trail solutions. These operations are repeated until the termination criteria satisfied.

The main objective of this paper is to construct an efficient algorithm which seeks optimal or near-optimal solutions of a given objective function for constrained problems by combining the genetic linear crossover with a DE algorithm to explore more solutions in the search space and to avoid trapping in local minima.

The reminder of this paper is organized as fellow. The problem definition and an overview of genetic algorithm and differential evolution are given in Section II. In Section III, we explain the proposed algorithm in detail. The numerical experimental results are presented in Section IV. Finally, the conclusion of the paper is presented in Section V.

PROBLEM FORMULATION

In the following section and subsections, we give an overview of the constraint optimization problem and we high light the penalty function technique, which are used to convert the constrained optimization problems to unconstrained optimization problems. Finally, we present the standard genetic algorithm and deferential evolutionary algorithm.

A. Constrained optimization problematic

A general form of a constrained optimization is defined as follows:

$$\text{Minimize } f(x),\ x = (x_1, x_2, \cdots, x_n)^T, \quad (1)$$
$$\text{Subject to}$$
$$g_i(x) \le 0,\ i = 1, \cdots, m$$
$$h_j(x) = 0,\ j = 1, \cdots, l$$
$$x_l \le x_i \le x_u$$

Where f (x) is the objective function, x is the vector of n variables, gi (x) ≤ 0 are inequality constraints, hj (x) = 0 are equality constraints, xl, xu are variables bounds. In this paper, we used the penalty function technique to solve constrained optimization problems. The following subsection gives more details about the penalty function technique.

1) The Penalty function method: The penalty function technique is used to transform the constrained optimization problems to unconstrained optimization problem by penalizing the constraints and forming a new objective function as follow:

$$f(x) = \begin{cases} f(x) & \text{if } x \in \text{feasible region} \\ f(x) + \text{penalty}(x) & x \in \text{feasible region.} \end{cases} \quad (2)$$

Where,

$$\text{penalty}(x) = \begin{cases} 0 \text{ if no constraint is violated} \\ 1 \text{ otherwise.} \end{cases}$$

There are two kinds of points in the search space of the constrained optimization problems (COP), feasible points which satisfy all constraints and unfeasible points which violate at least one of the constraints. At the feasible points, the penalty function value is equal the value of objective function, but at the infeasible points the penalty function value is equal to a high value as shown in Equation 2. In this paper, a non stationary penalty function has been used, which the values of the penalty function are dynamically changed during the search process. A general form of the penalty function as defined in [21] as follows:

$$F(x) = f(x) + h(k)H(x),\ x \in S \subset R^n, \quad (3)$$

Where f (x) is the objective function, h(k) is a non stationary (dynamically modified) penalty function, k is the current iteration number and H(x) is a penalty factor, which is calculated as follows:

$$H(x) = \sum_{i=1}^{m} \theta(q_i(x)) q_i(x)^{\gamma(q_i(x))} \quad (4)$$

Where qi (x) = max (0, gi (x)), i = 1, . . . , m, gi are the constrains of the problem, qi is a relative violated function of the constraints, θ(qi (x)) is the power of the penalty function, the values of the functions h(.), θ(.)and γ(.) are problem dependant. We applied the same values, which are reported in [21].

The following values are used for the penalty function:

$$\gamma(q_i(x)) = \begin{cases} 1 \text{ if } q_i(x) < 1, \\ 2 \text{ otherwise.} \end{cases}$$

Where the assignment function was

$$\theta(q_i(x))) = \begin{cases} 10 & \text{if } q_i(x) < 0.001, \\ 20 & \text{if } 0.001 \le q_i(x) < 0.1, \\ 100 & \text{if } 0.1 \le q_i(x) < 1, \\ 300 & \text{otherwise.} \end{cases}$$

And the penalty value h(t) = t * t.

B. Genetic algorithm overview

Genetic algorithm (GA) was introduced by Holland [8]. The basic principles of GA are inspired from the principles of life which were first described by Darwin [4]. GA starts with a number of individuals (chromosomes) which form a population. After randomly creating of the population, the initial population is evaluated using fitness function. The selection operator is starting to select highly fit individuals with high fitness function score to create new generation. Many type of selection have been developed like roulette wheel selection, tournament selection and rank selection [12]. The selected individuals are going to matting pool to generate offspring by applying crossover and mutation. Crossover operator is applied to the individuals in the mating pool to produces two new offspring from two parents by exchanging substrings. The most common crossover operators are one point crossover [8], two point crossover [12], and uniform crossover [12]. The parents are selected randomly in crossover operators by assign a random number to each parent, the parent with random number lower than or equal the probability of crossover ration Pc is always selected. Mutation operators are important for local search and to avoid premature convergence. The probability of mutation pm must be selected to be at a low level otherwise mutation would randomly change too many alleles and the new individual would have nothing in common with its parents. The new offspring is evaluated using fitness function, these operations are repeated until termination criteria stratified, for example number of iterations. The main structure of genetic algorithm is presented in Algorithm 1

Algorithm 1 genetic algorithm structure

1: Set the generation counter t := 0.
2: Generate an initial population P_0 randomly.
3: Evaluate the fitness function of all individuals in P_0.
4:
5: repeat
6: Set t = t + 1. { Generation counter increasing}.
Select an intermediate population P_t from P_{t-1}.
{Selection operator}.
7: Associate a random number r from (0, 1) with each row in P_t.
if $r < p_c$ then
8: Apply crossover operator to all selected pairs of P_t.
9: Update P_t.
10:
11: end if{Crossover operator}.
12: Associate a random number r_1 from (0, 1) with each gene in each individual in P_t.
if $r_1 < p_m$ then
13: Mutate the gene by generating a new random value
14: for the selected gene with its domain.
Update P_t.
end if
15: {Mutation operator}.
16: Evaluate the fitness function of all individuals in P_t.
until Termination criteria satisfied.

1) Liner crossover operator: HGDESCOP uses a linear crossover [20] in order to generate a new offspring to substitute their parents in the population. The main steps of the linear crossover are shown in Procedure 1.

Procedure 1: Linear Crossover (p1 , p2)

1. Generate three offspring $c_1 = (c_{11}, \ldots, c_{1D})$, $c_2 = (c_{21}, \ldots, c_{2D})$ and $c_3 = (c_{31}, \ldots, c_{3D})$ from parents $p_1 = (p_{11}, \ldots, p_{1D})$ and $p_2 = (p_{21}, \ldots, p_{2D})$, where

$$c_{1i} = \frac{p_{1i1} + p_{2i2}}{2},$$

$$c_{2i} = \frac{3p_{1i1} - p_{2i2}}{2},$$

$$c_{3i} = \frac{-p_{1i3} + p_{2i2}}{2},$$

$i = 1, \ldots, D.$

2. Choose the two most promising offspring of the three to substitute their parents in the population.
3. Return.

C. Differential evolution algorithm Differential evolution algorithm (DE) proposed by Stron and Price in 1997 [17]. In DE, the initial population consists of number of individuals, which is called a population size N P . Each individual in the population size is a vector consists of D dimensional variables and can be defined as follows:

$$x_i^{(G)} = \{x_{i,1}^{(G)}, x_{i,2}^{(G)}, \ldots, x_{i,D}^{(G)}\}, \quad i = 1, 2, \ldots, NP. \quad (5)$$

Where G is a generation number, D is a problem dimensional number and NP is a population size. DE employs mutation and crossover operators in order to generate a trail vectors, then the selection operator starts to select the individuals in new generation G+1. The overall process is presented in details as follows:

1) Mutation operator: Each vector xi in the population size creates a trail mutant vector vi as follows.

DE applied different strategies to generate a mutant vector as fellows

$$\text{DE/rand/1}: \quad v_i^{(G)} = x_{r1}^{(G)} + F \cdot (x_{r2}^{(G)} + x_{r3}^{(G)}) \quad (7)$$

$$\text{DE/best/1}: \quad v_i^{(G)} = x_{best}^{(G)} + F \cdot (x_{r1}^{(G)} + x_{r2}^{(G)}) \quad (8)$$

$$\text{DE/currenttobest/1}: \quad v_i^{(G)} = x_i^{(G)} + F \cdot (x_{best}^{(G)} - x_i^{(G)}) + F \cdot (x_{r1}^{(G)} - x_{r2}^{(G)}) \quad (9)$$

$$\text{DE/best/2}: \quad v_i^{(G)} = x_{best}^{(G)} + F \cdot (x_{r1}^{(G)} - x_{r2}^{(G)}) + F \cdot (x_{r3}^{(G)} - x_{r4}^{(G)}) \quad (10)$$

$$\text{DE/rand/2}: \quad v_i^{(G)} = x_{r1}^{(G)} + F \cdot (x_{r2}^{(G)} - x_{r3}^{(G)}) + F \cdot (x_{r4}^{(G)} - x_{r5}^{(G)}) \quad (11)$$

where rd , d = 1, 2, . . . , 5 represent random integer indexes, rd $\in$ [1, N P] and they are different from i. F is a mutation (G) scale factor, F $\in$ [0, 2]. xbest is the best vector in the population in the current generation G.

2) Crossover operator: A crossover operator starts after mutation in order to generate a trail vector according to target vector xi and mutant vector vi as follows:

$$u_{i,j} = \begin{cases} v_{i,j}, & \text{if rand}(0, 1) \le CR \text{ or } j = j_{rand} \\ x_{i,j}, & \text{otherwise} \end{cases} \quad (12)$$

Where CR is a crossover factor, CR $\in$ [0, 1] is a random integer and jrand $\in$ [0, 1]

3) Selection operator: The DE algorithm applied greedy selection, selects between the trails and targets vectors. The selected individual (solution) is the best vector with the better fitness value. The description of the selection operator is presented as fellows

$$x_i^{(G+1)} = \begin{cases} u_i^{(G)}, & \text{if } f(u_i) \le f(x_i^{(G)}) \\ x_i^{(G)}, & \text{otherwise} \end{cases}, \quad (13)$$

The main steps of DE algorithm are presented in Algorithm 2

Algorithm 2 The structure of differential evolution algorithm

```
Set the generation counter G := 0.
Set the initial value of F and CR.
Generate an initial population P 0 randomly.
Evaluate the fitness function of all individuals in P 0 .
repeat
   Set G = G + 1. {Generation counter increasing}.
   for i = 0; i < NP; i + + do
      Select random indexes r1 , r2 , r3 , where r1 = r2 =
      r3 = i.
                                        (G)(G)(G)(G)
   vi = xr1 + F × (xr2 − xr3 ). {Mutation oper-
      ator}.
      j = rand(1, D)
      for (k = 0; k < D; k + +) do
         if (rand(0, 1) ≤ CR or k = j then
                (G)(G)
            uik = vik {Crossover operator}
         else
                (G)(G)
            uik = xik
         end if
      end for
                (G)(G)
      if (f (ui ) ≤ f (xi )) then
            (G+1)(G)
         xi= ui {Greedy selection}.
      else
            (G+1)(G)
         xi= xi
      end if
   end for
until Termination criteria satisfied.
```

HGDESCOP ALGORITHM

HGDESCOP algorithm starts by setting the parameter values. In HGDESCOP, the initial population is generated randomly, which consists of NP individuals as shown in Equation 5. Each individual in the population is evaluated by using

the objective function. At each generation (G), each individual in the population is updated by applying the DE mutation operator by selecting a random three indexes r1 , r2 , r3 , where r1 = r2 = r3 = i as shown in Equations 6, 7. After updating the individual in the population, a random number r from (0, 1) is associated with each individual in the population by applying the genetic algorithm linear crossover operator as shown in Procedure 1. The greedy selection operator is starting to select the new individuals to form the new population in next generation as shown in Equation 13. These operations are repeated until termination criterion satisfied, which is the number of iterations in our algorithm.

Algorithm 3 The proposed HGDESCOP algorithm

1: Set the generation counter G := 0.
2: Set the initial value of F , p_c and NP.
3: Generate an initial population P_0 randomly.
4: Evaluate the fitness function of all individuals in P_0 .
5:
6: repeat
7: Set G = G + 1. {Generation counter increasing}.
8: for (i = 0; i < NP; i++) do
Select random indexes r_1 , r_2 , r_3 , where $r_1 = r_2 = r_3 = i$
$(G)(G)(G)(G)$
9: $v_i = x_{r1} + F \times (x_{r2} - x_{r3})$ {DE mutation operator}.
end for
10:
11: for (j = 0; j < NP; j++) do
12: Associate a random number r from (0, 1) with each $(G)v_j$ in $P_{(G)}$.
if r < P_c then
13:
14: Apply Procedure 1 to all selected pairs of v_i in $P_{(G)}$. {GA linear crossover operator}.
(G)
15: Update u_i .
16: end if
17: end for
18: for (k = 0; k < NP; k++) do
$(G)(G)$
19: if ($f(u_k) \le f(x_k)$) then
$(G+1)(G)$
20: $x_k = u_k$ {Greedy selection}.
21:
22: else
23: $(G+1)(G)$
24: $x_k = x_k$
25: end if
26: end for
Update $P_{(G)}$
until $Itr_{no} \le M$ axitr {Termination criteria satisfied}.

RESULTS OF NUMERICAL EXPERIMENTS

The general performance of the proposed HGDESCOP algorithm is tested using 13 benchmark function G1 − G13, which are reported in details in [5], [7], [13]. These functions are listed in Table I as follows.

TABLE I: Constrained benchmark functions.

F unction	D	Type of function	Optimal
G_1	12	quadratic	-13.000
G_2	19	nonlinear	-0.705619
G_3	11	polynomial	-1.1200
G_4	6	quadratic	-20465.539
G_5	5	cubic	4998.778
G_6	3	cubic	-6755.814
G_7	9	quadratic	22.662
G_8	3	nonlinear	-0.084558

TABLE II: HGDESCOP parameter settings.

Parameters	Definitions	Values
NP	Population size	29
p_c	Crossover probability	0.7
F	Mutation scale factor	0.6
M axitr	Maximum number of iterations	1000

A. Parameter settings

The parameters used by HGDESCOP and their values are summarized in Table II. These values are either based on the common setting in the literature or determined through our preliminary numerical experiments.

B. Performance analysis

In order to test the general performance of the proposed HGDESCOP algorithm, we applied it with 13 benchmark functions G1 − G13 and the results are reported in Table III.

Also, six functions have been plotted as shown in Figure 1.

1) The general performance of the HGDESCOP algorithm:

The best, mean, worst and standard deviation values are averaged over 30 runs and reported in Table III. We can observe from the results in Table III, that HGDESCOP could obtain the optimal solution or very near to optimal solution for all functions G1 − G12 for all 30 runs, However HGDESCOP could obtain the optimal solution with function G13 for 9 out of 30 runs. Also in Figure 1, we can observe that the function values are rapidly decreasing as the number of function generations increases.

We can conclude from Table III and Figure 1, that HGDE- SCOP is an efficient algorithm and it can obtain the optimal or near optimal solution with only few number of iterations.

C. HGDESCOP and other algorithms

In order to evaluate the performance of HGDESCOP algorithm, we compare it with four evolutionary based algorithms all results are reported in Table IV, and the results of the other algorithms are taken from their original papers. The four algorithms are listed as follows.

Homomorphous Mappings (HM) [9]
This algorithm, incorporates a homomorphous mapping between n-dimensional cube and a feasible search space.

- Stochastic Ranking (SR) [16]

This algorithm introduces a new method to balance objective and penalty functions stochastically, (stochastic ranking), and presents a new view on penalty function methods in terms of the dominance of penalty and objective functions.
Adaptive Segregational Constraint Handling EA (AS- CHEA) [6]
This algorithm is called ASCHEA and it is used after extending the penalty function and introducing niching techniques with adaptive radius to handel multimodel functions. The main idea of the algorithm is to start for each equality with a large feasible

TABLE III: Experimental results of HGDESCOP for $G_1 - G_8$

F unction	optimal	best	mean	worst	std
G_1	-15.000	-15.000	-15.000	-15.000	0.0e+00
G_2	-0.803619	-0.8036187	-0.7993549	-0.7861574	0.0062361
G_3	-1.000	-1.0005001	-1.0005000	-1.0004992	2.7368237e-07
G_4	-30665.539	-30665.538	-30665.538	-30665.538	0.0e+00
G_5	5126.498	5126.496858	5126.496728	5126.49671	4.5552e-05
G_6	-6961.814	-6961.813875	-6961.813875	-6961.813875	1.9173e-12
G_7	24.306	24.306209	24.306209	24.306209	4.706924e-13
G_8	-0.095825	-0.095825	-0.095825	-0.095825	1.223905e-17

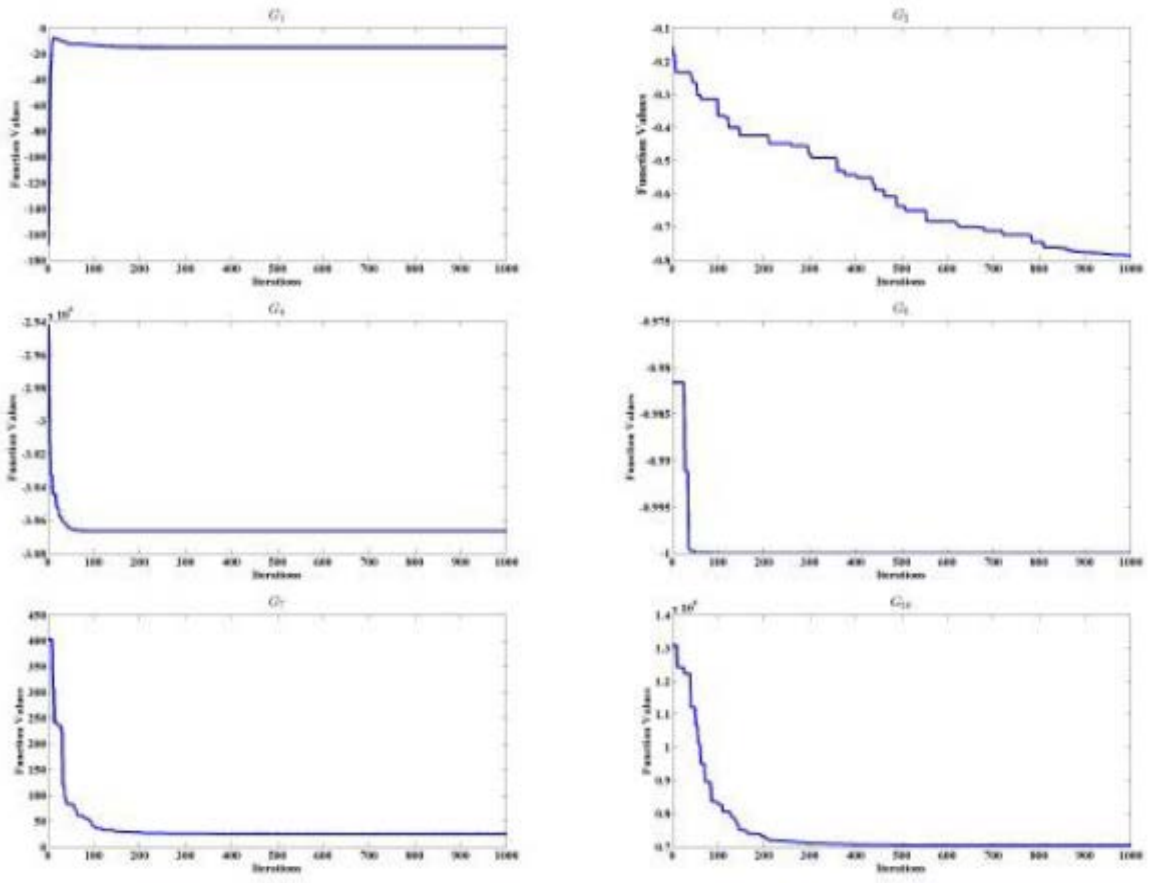

Fig. 1: The general performance of HGDESCOP algorithm.

Domain and to reduce it progressively in order to bring it as close as possible to null measure domain.

• Simple Multimembered Evolution Strategy (SMES)
[14].

This algorithm is based on a multimembered ES with a feasibility comparison mechanism.

1) Comparison between HM, SR, ASCHEA, SMES and
HGDESCOP: The best, mean, worst results of the five comparative algorithms are averaged over 30 runs and reported in Table IV. The evaluation function values for HM, SR, ASCHEA and SMES algorithms are 1,400,000, 350,000, 1,500,000 and 250,000 respectively. However the maximum evaluation function value for HGDESCOP algorithm is 120,000. We can observe from Table IV, that HGDE- SCOP results are better than the other algorithms for all functions G1 – G12 except the last function G13. In term of evaluation function values, it is clear that HGDESCOP is faster than the other algorithms.

CONCLUSION

In the paper the proposed algorithm (HGDESCOP) combines the differential evolution algorithm anv d the genetic linear crossover operator to improve the investigation ability of the DE algorithm and avoid trapping in local minima. To validate efficiency of proposed algorithm, it compared with 4 Evolutionary based algorithm on 8 benchmark functions. All results depict that HGDESCOP algorithm is best algorithm.

TABLE IV: Experimental results of HGDESCOP and other EA-based algorithms for problems G_1 – G_8

F unction	optimal		HM	SR	ASCHEA	SMES	HGDESCOP
G_1	-14.000	Best	-13.7864	-14	-14	-14	-14
	-14.000	Mean	-137082	-14	-13.84	-14	-14
	-14.000	Worst	-13.6154	-14	N.A.	-14	-14
G_2	-0.702619	Best	0.68953	0.799515	0.785	0.703601	-0.7936187
	-0.703619	Mean	0.68671	0.699975	0.59	0.691322	-0.7793549
	-0.708619	Worst	0.68119	0.631288	N.A.	0.661322	-0.7661574
G_3	-1.000	Best	0.9997	1.000	1.000	1.001038	1.000500
	-1.000	Mean	0.9989	1.000	0.99989	1.000989	1.0005000
	-1.000	Worst	0.9978	1.000	N.A.	1.000579	1.0004992
G_4	-29665.539	Best	-30664.5	-30665.539	-30665.5	-30665.539062	-30665.538
	-29665.539	Mean	-30655.3	-30665.539	-30665.5	-30665.539062	-30665.538
	-29665.539	Worst	-30645.9	-30665.539	N.A.	-30665.539062	-30665.538
G_5	4926.498	Best	-	5026.497	5126.5	5126.599609	5126.496728
	4926.498	Mean	-	5028.881	5141.65	5174.492301	5226.496728
	4926.498	Worst	-	5042.472	N.A.	5304.166992	5026.49671
G_6	-6761.814	Best	-6752.1	-6861.814	-6961.81	-6861.813965	-6761.813875
	-6761.814	Mean	-6142.6	-6775.940	-6961.81	-6681.283984	-6761.813875
	-6761.814	Worst	-5373.9	-6250.262	N.A.	-6861.481934	-6761.813875
G_7	24.306	Best	24.620	24.307	24.3323	24.326715	24.306209
	24.306	Mean	24.826	24.374	24.6636	24.474926	24.306209
	24.306	Worst	25.069	24.842	N.A.	24.842829	24.306209
G_8	0.095825	Best	0.0958250	0.095825	0.09582	0.095826	0.095825
	0.095825	Mean	0.0891568	0.095825	0.09582	0.095826	0.095825
	0.095825	Worst	0.0291438	0.095825	N.A.	0.095826	0.095825

And achieve the global minima or near global minima earlier than other algorithms. Results also depicts that accuracy, reliability using (HGDESCOP) is much better than defined methods

REFERENCES

1. A.Igelais, "Proofs of Correctness of Data Representations," Acta Informatica, pp-56,vol. 1, no. 4 (2006).
2. C.James, "Monitors: An Operating System Structuring Concept,"Communications of Information system,pp-193,vol. 5, no7, 2005.
3. P. Caamano, F. Bellas, J. A. Becerra, and R. J. Duro, Evolutionary algorithm characterization in real parameter optimization problems, Applied Soft Computing, vol. 13, no. 4, pp. 1902-1921, 2013.
4. C. Darwin, On the Origin of Species. London: John Murray, 1859.
5. Codes. In Lecture notes in economics and mathematical systems (Vol. 187). Berlin: Springer, 1981.
6. J. H. Holland, Adaptation in Natural and Artficial Systems, University of Michigan Press, Ann Arbor, MI, 1975. S. Koziel and Z. Michalewicz, Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization, Evolutionary Computation 7(1), 19-44, 1999.
7. T.S. Metcalfe, P. Charbonneau, Stellar structure modeling using a parallel genetic algorithm for objective global optimization, Journal of Computational Physics 185, 176-193, 2003.
8. Z. Michalewicz, A Survey of Constraint Handling Techniques in Evo- lutionary Computation Methods, Evolutionary Programming, Vol.4, pp. 135, 1995.Z. Michalewicz, Genetic algorithms + data structures = evolution programs,Berlin,Springer,1996
9. Z. Michalewicz and M. Schoenauer, Evolutionary algorithms for con- strained parameter optimization problems, Evolutionary Computation 4(1), 132, 1996. E. M. Montes and C. A. Coello Coello, A simple multi-membered
10. Evolution strategy to solve constrained optimization problems, IEEE Transactions on Evolutionary Computation, vol. 9, no. 1, pp. 117, 2005. Y. F. Ren and Y. Wu, An efficient algorithm for high-dimensional
11. Function optimization, Soft Computing, vol. 17, no. 6, pp. 995-1004, 2013. T.P. Runarsson and X. Yao, Stochastic Ranking for Constrained Evolutionary Optimization, IEEE Transactions on Evolutionary Computation4 (3), 284-294, 2000.
12. R. Storn, K. Price, Differential evolutiona simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11:341- 359, 1997.
13. Y. Wang and C. Zixing, A hybrid multi-swarm particle swarm optimization to solve constrained optimization problems, Frontiers of Computer Science in China, Vol. 3, No. 1, pp. 38-52, March, 2009.
14. Y. Wang, Zixing Cai and Yuren Zhou. Accelerating adaptive trade-off model using shrinking space technique for constrained evolutionary optimization, International Journal for Numerical Methods in Engineering, Vol. 77, No. 11, pp. 1501-1534, March 2009.