A novel hybrid genetic differential evolution algorithm for constrained optimization problems

Ahmed Fouad Ali
Faculty of Computers and Information
Dept. of Computer Science, Suez Canal University
Ismailia, Egypt
Email: ahmed_fouad@ci.suez.edu.eg

***Abstract*—Most of the real-life applications have many constraints and they are considered as constrained optimization problems (COPs). In this paper, we present a new hybrid genetic differential evolution algorithm to solve constrained optimization problems. The proposed algorithm is called hybrid genetic differential evolution algorithm for solving constrained optimization problems (HGDESCOP). The main purpose of the proposed algorithm is to improve the global search ability of the DE algorithm 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. In order to verify the general performance of the HGDESCOP algorithm, it has been compared with 4 evolutionary based algorithms on 13 benchmark functions. The experimental results show that the HGDESCOP algorithm is a promising algorithm and it outperforms other algorithms.**

***Keywords*—Constrained optimization problems, Genetic algorithms, Differential evolution algorithm, Linear crossover.**

I. Introduction

Evolutionary algorithms (EAs) have been widely used to solve many unconstrained optimization problems [1], [3], [10], [15]. EAs are unconstrained search algorithms and lake a technique to handel the constraints in the constrained optimization problems (COPs). There are different techniques to handle constraints in EAs, these techniques have been classified by Michalewicz [13] 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 (DE) is one of the most widely used evolutionary algorithms (EAs) introduced by Stron and Price [17]. 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) [2], [18], [19]. 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 the objective function. At each generations, the new offspring is created by applying the DE mutation. In order to increase the global search behavior 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.

II. Problem Definition and Overview of genetic algorithm and differential evolution algorithm

In the following section and subsections, we give an overview of the constraint optimization problem and we highlight 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 problems

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

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

Where $f(x)$ is the objective function, x is the vector of n variables, $g_i(x) \le 0$ are inequality constraints, $h_j(x) = 0$ are equality constraints, x_l, x_u are variables bounds. In this paper, we used the penalty function technique to solve constrained optimization problems [11]. The following subsection gives more details about the penalty function technique.

1) The Penalty function technique: 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:

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

Where,

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

There are two kind 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:

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

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 $q_i(x) = \max(0, g_i(x)), i = 1, \ldots, m$, g_i are the constrains of the problem, q_i is a relative violated function of the constraints, $\theta(q_i(x))$ is the power of the penalty function, the values of the functions $h(.), \theta(.)\text{and}\gamma(.)$ 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 $h(t) = t * \sqrt{t}$.

B. An overview of genetic algorithm

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 start 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], 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 P_c is always selected. Mutation operators are important for local search and to avoid premature convergence. The probability of mutation p_m 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 The structure of genetic algorithm

```
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:  repeat
5:      Set $t = t + 1$. { Generation counter increasing}.
6:      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 $P^t$.
8:      if $r < p_c$ then
9:          Apply crossover operator to all selected pairs of $P^t$.
10:         Update $P^t$.
11:     end if{Crossover operator}.
12:     Associate a random number $r_1$ from $(0, 1)$ with each
        gene in each individual in $P^t$.
13:     if $r_1 < p_m$ then
14:         Mutate the gene by generating a new random value
            for the selected gene with its domain.
15:         Update $P^t$.
16:     end if
        {Mutation operator}.
17:     Evaluate the fitness function of all individuals in $P^t$.
18: 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 is shown in Procedure 1.

Procedure 1: Linear Crossover(p^1, p^2)

1. Generate three offspring $c^1 = (c^1_1, \ldots, c^1_D)$, $c^2 = (c^2_1, \ldots, c^2_D)$ and $c^3 = (c^3_1, \ldots, c^3_D)$ from parents

$p^1 = (p^1_1, \ldots, p^1_D)$ and $p^2 = (p^2_1, \ldots, p^2_D)$, where

$$\begin{aligned} c^1_i &= \frac{1}{2}p^1_i + \frac{1}{2}p^2_i, \\ c^2_i &= \frac{3}{2}p^1_i - \frac{1}{2}p^2_i, \\ c^3_i &= -\frac{1}{2}p^1_i + \frac{3}{2}p^2_i, \end{aligned}$$

$i = 1, \ldots, D$.

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

C. An overview of 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 NP. Each individual in the population size is a vector consists of D dimensional variables and can be defined as follows:

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

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 $\mathbf{x}_i$ in the population size create a trail mutant vector $\mathbf{v}_i$ as follows.

$$\mathbf{v}_i^{(G)} = \{v_{i,1}^{(G)}, v_{i,2}^{(G)}, \ldots, x_{i,D}^{(G)}\}, \quad i = 1, 2, \ldots, NP. \tag{6}$$

DE applied different strategies to generate a mutant vector as fellows:

$$\begin{aligned} DE/rand/1: \quad & \mathbf{v}_i^{(G)} = \mathbf{x}_{r_1}^{(G)} + F \cdot (\mathbf{x}_{r_2} + \mathbf{x}_{r_3}) && (7) \\ DE/best/1: \quad & \mathbf{v}_i^{(G)} = \mathbf{x}_{best}^{(G)} + F \cdot (\mathbf{x}_{r_1} + \mathbf{x}_{r_2}) && (8) \\ DE/currenttobest/1: \quad & \mathbf{v}_i^{(G)} = \mathbf{x}_i^{(G)} + F \cdot (\mathbf{x}_{best} - \mathbf{x}_i) \\ & \quad + F \cdot (\mathbf{x}_{r_1} - \mathbf{x}_{r_2}) && (9) \\ DE/best/2: \quad & \mathbf{v}_i^{(G)} = \mathbf{x}_{best}^{(G)} + F \cdot (\mathbf{x}_{r_1} - \mathbf{x}_{r_2}) \\ & \quad + F \cdot (\mathbf{x}_{r_3} - \mathbf{x}_{r_4}) && (10) \\ DE/rand/2: \quad & \mathbf{v}_i^{(G)} = \mathbf{x}_{r_1}^{(G)} + F \cdot (\mathbf{x}_{r_2} - \mathbf{x}_{r_3}) \\ & \quad + F \cdot (\mathbf{x}_{r_4} - \mathbf{x}_{r_5}) && (11) \end{aligned}$$

where r_d, $d = 1, 2, \ldots, 5$ represent random integer indexes, $r_d \in [1, NP]$ and they are different from i. F is a mutation scale factor, $F \in [0, 2]$. $\mathbf{x}_{best}^{(G)}$ 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 $\mathbf{x}_i$ and mutant vector $\mathbf{v}_i$ as follows:

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

Where *CR* is a crossover factor, *CR* $\in [0, 1]$, j_{rand} is a random integer and $j_{rand} \in [0, 1]$

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:

$$\mathbf{x}_i^{(G+1)} = \begin{cases} \mathbf{u}_i^{(G)}, & \text{if} \quad f(\mathbf{u}_i^{(G)}) \le f(\mathbf{x}_i^{(G)}), \\ \mathbf{x}_i, & \text{otherwise} \end{cases} \tag{13}$$

The main steps of DE algorithm are presented in Algorithm 2

Algorithm 2 The structure of differential evolution algorithm

```
1: Set the generation counter $G := 0$.
2: Set the initial value of $F$ and CR.
3: Generate an initial population $P^0$ randomly.
4: Evaluate the fitness function of all individuals in $P^0$.
5: repeat
6:   Set $G = G + 1$. {Generation counter increasing}.
7:   for $i = 0; i < NP; i++$ do
8:     Select random indexes $r_1$, $r_2$, $r_3$, where $r_1 \neq r_2 \neq r_3 \neq i$.
9:     $\mathbf{v}_i^{(G)} = \mathbf{x}_{r_1}^{(G)} + F \times (\mathbf{x}_{r_2}^{(G)} - \mathbf{x}_{r_3}^{(G)})$. {Mutation operator}.
10:    $j = rand(1, D)$
11:    for $(k = 0; k < D; k++)$ do
12:      if $(rand(0,1) \le CR$ or $k = j$ then
13:        $u_{ik}^{(G)} = v_{ik}^{(G)}$ {Crossover operator}
14:      else
15:        $u_{ik}^{(G)} = x_{ik}^{(G)}$
16:      end if
17:    end for
18:    if $(f(\mathbf{u}_i^{(G)}) \le f(\mathbf{x}_i^{(G)}))$ then
19:      $\mathbf{x}_i^{(G+1)} = \mathbf{u}_i^{(G)}$ {Greedy selection}.
20:    else
21:      $\mathbf{x}_i^{(G+1)} = \mathbf{x}_i^{(G)}$
22:    end if
23:  end for
24: until Termination criteria satisfied.
```

III. The proposed 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 r_1, r_2, r_3, where $r_1 \neq r_2 \neq r_3 \neq i$ 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:  repeat
6:      Set $G = G + 1$. {Generation counter increasing}.
7:      for ($i = 0; i < NP; i{+}{+}$) do
8:          Select random indexes $r_1$, $r_2$, $r_3$, where $r_1 \neq r_2 \neq r_3 \neq i$
9:          $\mathbf{v}_i^{(G)} = \mathbf{x}_{r_1}^{(G)} + F \times (\mathbf{x}_{r_2}^{(G)} - \mathbf{x}_{r_3}^{(G)})$ {DE mutation operator}.
10:     end for
11:     for ($j = 0; j < NP; j{+}{+}$) do
12:         Associate a random number $r$ from $(0, 1)$ with each $\mathbf{v}_j^{(G)}$ in $P^{(G)}$.
13:         if $r < P_c$ then
14:             Apply Procedure 1 to all selected pairs of $\mathbf{v}_i^{(G)}$ in $P^{(G)}$. {GA linear crossover operator}.
15:             Update $\mathbf{u}_i^{(G)}$.
16:         end if
17:     end for
18:     for ($k = 0; k < NP; k{+}{+}$) do
19:         if ($f(\mathbf{u}_k^{(G)}) \leq f(\mathbf{x}_k^{(G)})$) then
20:             $\mathbf{x}_k^{(G+1)} = \mathbf{u}_k^{(G)}$ {Greedy selection}.
21:         else
22:             $\mathbf{x}_k^{(G+1)} = \mathbf{x}_k^{(G)}$
23:         end if
24:     end for
25:     Update $P^{(G)}$
26: until $Itr_{no} \leq Maxitr$ {Termination criteria satisfied}.
```

IV. Numerical experiments

The general performance of the proposed HGDESCOP algorithm is tested using 13 benchmark function $G_1 - G_{13}$, which are reported in details in [5], [7], [13]. These functions are listed in Table I as follows.

TABLE I: Constrained benchmark functions.

$Function$	D	Type of function	Optimal
G_1	13	quadratic	-15.000
G_2	20	nonlinear	-0.803619
G_3	10	polynomial	-1.000
G_4	5	quadratic	-30665.539
G_5	4	cubic	5126.498
G_6	2	cubic	-6961.814
G_7	10	quadratic	24.306
G_8	2	nonlinear	-0.095825
G_9	7	polynomial	680.630
G_{10}	8	linear	7049.248
G_{11}	2	quadratic	0.75
G_{12}	3	quadratic	-1.000
G_{13}	5	nonlinear	0.053950

TABLE II: HGDESCOP parameter settings.

Parameters	Definitions	Values
NP	Population size	30
p_c	Crossover probability	0.8
F	Mutation scale factor	0.7
$Maxitr$	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 $G_1 - G_{13}$ 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 $G_1 - G_{12}$ for all 30 runs, However HGDESCOP could obtain the optimal solution with function G_{13} for 9 out of 30 runs. Also in Figure 1, we can observe that the function values are rapidly decrease as the number of function generations increases.

We can conclude from Table III and Figure 1, that HGDESCOP 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 (ASCHEA) [6]
  This algorithm is called ASCHEA and it is used after extending the penalty function and introducing a 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 $G_1 - G_{13}$

$Function$	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	$1.223905e^{-17}$
G_9	680.630	680.630057	680.630057	680.630057	$3.789561e^{-14}$
G_{10}	7049.248	7049.248020	7049.248020	7049.248020	$6.264592e^{-12}$
G_{11}	0.75	0.749900	0.749900	0.749900	$1.170277e^{-16}$
G_{12}	-1.000	-1.000	-1.000	-1.000	0.0e+00
G_{13}	0.053950	0.084356	0.372933	0.438802	0.139366

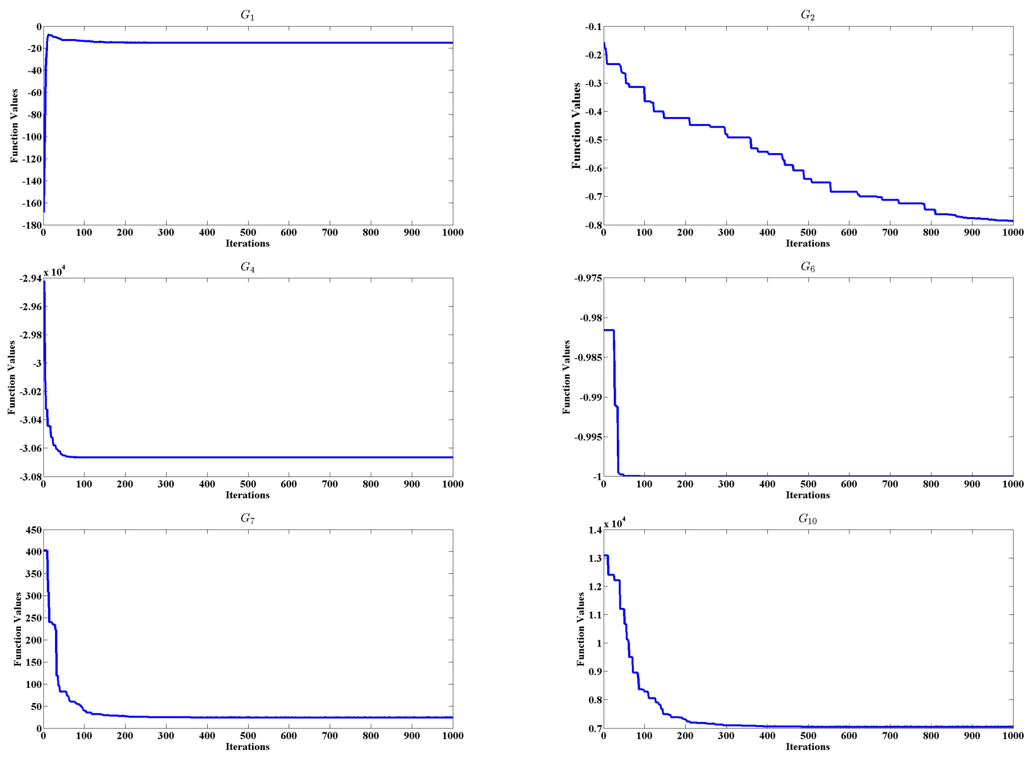

Fig. 1: The general performance of HGDESCOP algorithm.

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 HGDESCOP results are better than the other algorithms for all functions $G_1 - G_{12}$ except the last function G_{13}. In term of evaluation function values, it is clear that HGDESCOP is faster than the other algorithms.

V. Conclusion

In this paper, a new hybrid genetic differential evolution algorithm to solve constrained optimization problems is presented. The proposed algorithm is called hybrid genetic differential evolution algorithm for solving constrained optimization problems (HGDESCOP). The proposed algorithm combines the differential evolution algorithm and the genetic linear crossover operator in order to improve the exploration ability of the DE algorithm and to avoid trapping in local minima. To verify the efficiency of the proposed algorithm, it has been compared with 4 Evolutionary based algorithm on 13 benchmark functions. The experimental results show that the HGDESCOP algorithm is a robust and efficient algorithm

TABLE IV: Experimental results of HGDESCOP and other EA-based algorithms for problems $G_1 - G_{13}$

$Function$	optimal		HM	SR	ASCHEA	SMES	HGDESCOP
G_1	-15.000	Best	-14.7864	-15	-15	-15	-15
	-15.000	Mean	-14.7082	-15	-14.84	-15	-15
	-15.000	Worst	-14.6154	-15	N.A.	-15	-15
G_2	-0.803619	Best	0.79953	0.803515	0.785	0.803601	-0.8036187
	-0.803619	Mean	0.79671	0.781975	0.59	0.751322	-0.7993549
	-0.803619	Worst	0.79119	0.726288	N.A.	0.751322	-0.7861574
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	-30665.539	Best	-30664.5	-30665.539	-30665.5	-30665.539062	-30665.538
	-30665.539	Mean	-30655.3	-30665.539	-30665.5	-30665.539062	-30665.538
	-30665.539	Worst	-30645.9	-30665.539	N.A.	-30665.539062	-30665.538
G_5	5126.498	Best	-	5126.497	5126.5	5126.599609	5126.496728
	5126.498	Mean	-	5128.881	5141.65	5174.492301	5126.496728
	5126.498	Worst	-	5142.472	N.A.	5304.166992	5126.49671
G_6	-6961.814	Best	-6952.1	-6961.814	-6961.81	-6961.813965	-6961.813875
	-6961.814	Mean	-6342.6	-6875.940	-6961.81	-6961.283984	-6961.813875
	-6961.814	Worst	-5473.9	-6350.262	N.A.	-6961.481934	-6961.813875
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	24.642	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
G_9	680.630	Best	680.91	680.630	680.630	680.631592	680.630057
	680.630	Mean	681.16	680.656	680.641	680.643410	680.630057
	680.630	Worst	683.18	680.763	N.A.	680.719299	680.630057
G_{10}	7049.248	Best	7147.9	7054.316	7061.13	7051.902832	7049.248020
	7049.248	Mean	8163.6	7559.192	7497.434	7253.047005	7049.248020
	7049.248	Worst	9659.3	8835.655	N.A.	7638.366211	7049.248020
G_{11}	0.75	Best	0.75	0.750	0.75	0.749090	0.749900
	0.75	Mean	0.75	0.750	0.75	0.749358	0.749900
	0.75	Worst	0.75	0.75	N.A.	0.749830	0.749900
G_{12}	-1.000	Best	-0.999999875	-1.00000	N.A	-1.000000	-1.000000
	-1.000	Mean	-0.999134613	-1.00000	N.A.	-1.00000	-1.00000
	-1.000	Worst	-0.991950498	-1.00000	N.A.	-1.00000	-1.00000
G_{13}	0.053950	Best	N.A.	0.053957	N.A	0.053986	0.084356
	0.053950	Mean	N.A.	0.057006	N.A	0.166385	0.372933
	0.053950	Worst	N.A	0.216915	N.A	0.468294	0.438802

and can obtain the global minima or near global minima faster than other algorithms. AS part of our future work, in this paper we are using linear crossover to improve the performance of DE, whether the HGDESCOP algorithm could be improved by using more advanced GA crossover operators. Also we can apply the proposed algorithm with many real-life applications such as engineering design, finance, economics.

REFERENCES

[1] A.F. Ali, A. E. Hassanien, Minimizing molecular potential energy function using genetic Nelder-Mead algorithm, 8th International Conference on Computer Engineering & Systems (ICCES), Cairo, pp. 177-183, 2013.

[2] M.M. Ali and W.X. Zhu. A penalty function-based differential evolution algorithm for constrained global optimization, Computational Optimization and Applications, Vol. 54, No. 3, pp. 707–739, April 2013.

[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] C.A. Floudas and P.M. Pardalos, A collection of test problems for constrained global optimization algorithms. In P. M. Floudas (Ed.), Lecture notes in computer science, Vol. 455. Berlin: Springer, 1987.

[6] S. B. Hamida and M. Schoenauer, ASCHEA: New rsults using adaptive segregational constraint handling, in Proceedings of the Congress on Evolutionary Computation (CEC2002), Piscataway, New Jersey, IEEE Service Center, pp. 884-889, 2002.

[7] W. Hock, K.Schittkowski, Test examples for nonlinear programming codes. In Lecture notes in economics and mathematical systems (Vol. 187). Berlin: Springer, 1981.

[8] J. H. Holland, Adaptation in Natural and Artficial Systems, University of Michigan Press, Ann Arbor, MI, 1975.

[9] S. Koziel and Z. Michalewicz, Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization, Evolutionary Computation 7(1), 19-44, 1999.

[10] 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.

[11] Z. Michalewicz, A Survey of Constraint Handling Techniques in Evolutionary Computation Methods, Evolutionary Programming, Vol.4, pp. 135, 1995.

[12] Z. Michalewicz, Genetic algorithms + data structures = evolution

programs. Berlin: Springer, 1996.

[13] Z. Michalewicz and M. Schoenauer, Evolutionary algorithms for constrained parameter optimization problems, Evolutionary Computation 4(1), 132, 1996.

[14] E. M. Montes and C. A. Coello Coello, A simple multi-membered evolution strategy to solve constrained optimization problems, IEEE Transactions on Evolutionary Computation, vol. 9, no. 1, pp. 117, 2005.

[15] Y. F. Ren and Y. Wu, An efficient algorithm for high-dimensional function optimization, Soft Computing, vol. 17, no. 6, pp. 995-1004, 2013.

[16] T.P. Runarsson and X. Yao, Stochastic Ranking for Constrained Evolutionary Optimization, IEEE Transactions on Evolutionary Computation 4(3), 284-294, 2000.

[17] R. Storn, K. Price, Differential evolutiona simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11:341-359, 1997.

[18] 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.

[19] 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.

[20] A. Wright, Genetic Algorithms for Real Parameter Optimization, Foundations of Genetic Algorithms 1, G.J.E Rawlin (Ed.) (Morgan Kaufmann, San Mateo), 205-218, 1991.

[21] J.M. Yang, Y.P. Chen, J.T. Horng and C.Y. Kao. Applying family competition to evolution strategies for constrained optimization. In Lecture Notes in Mathematics Vol. 1213, pp. 201-211, New York, Springer, 1997.