Hybrid Constrained Evolutionary Algorithm for Numerical Optimization Problems

Wali Khan Mashwani∗†, Alam Zaib‡, Özgür Yeniay§, Habib Shah¶, Naseer Mansoor Tairan∥ and Muhammad Sulaiman∗∗

Abstract

Constrained optimization are naturally arises in many real-life applications, and is therefore gaining a constantly growing attention of the researchers. Evolutionary algorithms are not directly applied on constrained optimization problems. However, different constraint-handling techniques are incorporated in their framework to adopt it for dealing with constrained environments. This paper suggests an hybrid constrained evolutionary algorithm (HCEA) that employs two penalty functions simultaneously. The suggested HCEA has two versions namely HCEA-static and HCEA-adaptive. The performance of the HCEA-static and HCEA-adaptive algorithms are examined upon the constrained benchmark functions that are recently designed for the special session of the 2006 IEEE Conference of Evolutionary Computation (IEEE-CEC’06). The experimental results of the suggested algorithms are much promising as compared to one of the recent constrained version of the JADE. The converging behaviour of the both suggested algorithms on each benchmark function is encouraging and promising in most cases.

Keywords: Constrained Functions, Evolutionary Computation(EC), Evolutionary Algorithm(EA) and Hybrid EAs.

Received : 14.04.2017 Accepted : 04.09.2018 Doi : 10.15672/HJMS.2018.625

* Kohat University of Science & Technology, KPK, Pakistan, Email: mashwanigr8@gmail.com
† Corresponding Author.
‡ Kohat University of Science & Technology, KPK, Pakistan, Email: alamzaibmaths@gmail.com
§ Hacettepe University, Ankara, Turkey, Email: yeniay@hacettepe.edu.tr
¶ King Khalid University Abha, Saudi Arabia, Email: habibehah.uthm@gmail.com
∥ King Khalid University Abha, Saudi Arabia, Email: nmtairan@kku.edu.sa
∗∗ Abdul Wali Khan University, Mardan, Pakistan, Email: msulaiman@awkum.edu.pk
1. Introduction

In the last two decades and so, numerical optimization has become an emerging area of research because of their wide application in different discipline of sciences and engineering [12, 11]. The optimization problems have many types including multi-quadratic programming, bilinear and biconvex, generalized geometric programming, general constrained nonlinear optimization, bilevel optimization, complementarity, semidefinite programming, mixed-integer nonlinear optimization, combinatorial optimization and optimal control problems [19]. Generally these problems can be categorized into constrained and unconstrained optimization problems. In this paper, we are interested in solving optimization problems with continuous variables. In this paper, we are interested in solving the constrained optimization problems that can generally formulated as follow [14]:

\[
\begin{align*}
\text{Minimize} & \quad f(x), \quad x \in S \\
\text{subject to} & \quad g_i(x) \leq 0, \quad i = 1, 2, 3, \ldots, p; \\
& \quad h_j(x) = 0, \quad j = 1, 2, 3, \ldots, q,
\end{align*}
\]

where \( S \) denotes the whole search space, \( p \) is the number of inequality constraints and \( q \) is the number of equality constraints. If a problem at hand has some equality constraints, they can transform into inequalities as follow:

\[
(1.2) \quad h_j - \epsilon \leq 0
\]

where \( \epsilon \) is the tolerance rate. The inequality constraints that satisfy \( g_i(x) = 0 \) are called active constraints, it is also to be noted that equality constraints are always active.

In the last two decades and so, very many optimization methods developed in the form of deterministic and stochastic natures. Deterministic approaches involve no randomness to perform their search process. Interval optimization [13], branch-and-bound [27, 55] and algebraic techniques [54] are commonly used deterministic methods. On the other hand, stochastic nature based algorithms evolve their set of solutions with randomness. Simulated annealing (SA) [23, 22], Monte Carlo sampling [15], stochastic tunneling [32], parallel tempering [34], Genetic Algorithm (GA) [17], Evolutionary Strategies (ES) [43], Evolutionary Programming (EP) [20, 21, 7], Particle Swarm Optimization (PSO) [25, 67], Ant Colony Optimization (ACO) [60], differential evolution (DE) [52], Krill herd algorithm based on cuckoo search [2, 62], Elephant Herding Optimization (EHO) [8, 58], Moth search algorithm [56], Monarch Butterfly Optimization (MBO) [57], Earthworm Optimization Algorithm (EWA) [59], Plant Prorogation Algorithm (PPA) [49, 50, 51, 44] and hybrid EAs [28, 47, 18] are well-known stochastic methods. Evolutionary computation is the collective name used for population base evolutionary algorithms. These algorithms are mainly inspired by biological process of evolution, such as natural selection and genetic inheritance [16].

In general, evolutionary algorithms employ penalty functions and other constraint handling techniques to maintain a reasonable ratio among feasible and infeasible solutions for dealing with constrained optimization problems [33, 64, 63, 65, 9, 36]. Penalty functions are very common and popular approaches while adopting unconstrained EAs to constrained one. Most of the Researchers prefer adaptive penalty methods in order to develop constrained EAs to handle complex COPs. Recently, hybrid evolutionary algorithms have got much attention due to their high potentialities and capabilities to solve problems with high complexity, noisy environment, imprecision, uncertainty and vagueness [35, 1]. In this paper, We have combined two popular EAs including the PSO and differential evolution (DE) and developed hybrid constrained evolutionary algorithm to handle test problems designed for the special session of the 2006 IEEE-congress on evolutionary computation (IEEE-CEC’06) [31]. The suggested hybrid constrained EA
utilizes two penalty functions simultaneously. The suggested algorithms have tackled most of used test problems an affective manner.

The rest of the paper is organized as follows. Section 2 presents the framework of the proposed hybrid constrained evolutionary algorithm. Section 3 demonstrates experimental results. Section 4 concludes this research paper with future plan and directions.

2. Hybrid Constrained Evolutionary Algorithm

Evolutionary algorithms (EAs) have gained popularity and much attention of the researchers in academia and industrial applications. They have tackled various optimization and search problems comprising various complexities like noisy environments, imprecision, uncertainty and vagueness in their mathematical structures. EAs operate on set of solutions called population and ultimately they provide a set of optimal solutions in single simulation run. They do not require any derivative information regarding the objective function as well as constraint functions of the problems at hand. They use various intrinsic evolutionary operators like reproduction, mutation, recombination and selection to perform their search process.

In the last two decades and so, hybrid evolutionary algorithms (EAs) have got much attention for dealing with optimization problems with high complexity, noisy environment, imprecision, uncertainty and vagueness [1, 6, 42, 37, 38, 40, 4, 5, 61, 3, 41, 26, 45, 46]. In this paper, We have developed an efficient constrained hybrid constrained EAs by incorporating some existing penalty functions with static and self-adaptive procedures. The suggested HCEAs employs Differential evolution (DE) [52] and Particle Swarm Optimization (PSO) [25] as constituent search operators to perform their search process. The suggested algorithm also employs the penalty functions to improve the quality of feasible solutions. The penalty functions as given in equation 2.1 adopted with static and penalty function as explained in 2.2-2.8 are employed adaptive procedure in the framework of the Algorithm (1).

2.1. Penalty Functions. In the last decades, various Penalty Functions developed and found in the existing literature of the evolutionary computing [35, 33, 64, 63, 65, 9, 36]. These penalty functions are used to penalize the candidate solutions that violate the constraint functions of the problem 1.1. The first penalty function which was proposed in [24] that defines different levels of violation keeping in view the magnitude of violation of the constraint functions. Penalty Function works as follow:

- Define $l$ levels of violation for each constraint.
- Generate penalty coefficient $R_{ij}$, where $i = 1, \ldots, l$ and $j = 1, \ldots, m$ for each level of violation and each constraint. The bigger coefficients are given to the bigger violation levels.
- Generate a random population using both feasible and unfeasible individuals.
- Evaluate these individuals by using following formula

$$eval(\mathbf{x}) = f(\mathbf{x}) + \sum_{j=1}^{m} R_{ij} \max[0, g_j(\mathbf{x})]^2$$

(2.1)

where $R_{ij}$ denoted the penalty coefficient with respect to $j^{th}$ constraint and $i^{th}$ violation level and $m$ is the number of constraints. Homaifar et al. [24] transformed equality constraints to inequality constraints according to $|h_j(\mathbf{x})| - \epsilon \leq 0$, where $\epsilon$ is a small positive number. Adjustment of large number of parameters settings such as $m(2l + 1)$ is one of the main issue with static penalty functions proposed in [24]. For example, if $m = 5$ and $l = 4$ levels of violation then one has to adjust 45 parameters at same time. Although, complexity of this strategy is very high but still quite useful strategy
Algorithm 1 Framework of the Hybrid Constrained Evolutionary Algorithm

1: $N =$ Population Size,
2: $n =$ Dimension of the Search Space,
3: $X = \{x^1, \ldots, x^N\}^T \leftarrow \text{Initialize-Population}(N, n)$
4: $F = \{f(x^1), \ldots, f(x^N)\} \leftarrow \text{Evaluate}(\{x^1, \ldots, x^N\}^T)$
5: $G = \{g(x^1), \ldots, g(x^N)\} \leftarrow \text{Evaluate}(\{x^1, \ldots, x^N\}^T)$
6: \textbf{Apply the penalty} % (i.e., For Static Penalty Function referred to algorithm (2), or for Adaptive Penalty Function referred to algorithm (3)).
7: \textbf{for} $i \leftarrow 1 : N$ \textbf{do}
8: \hspace{1em} if $\text{rand} < 0.15$ then
9: \hspace{2em} Select $x^i, x^{i1}, x^{i2}$ at random from $X$ such that $x^i \neq x^{i1} \neq x^{i2}$
10: \hspace{2em} $u^i = x^i + F(x^{i1} - x^{i2})$
11: \hspace{2em} \textbf{Apply the penalty} % (i.e., For Static Penalty Function referred to algorithm (2), or for Adaptive Penalty Function referred to algorithm (3)).
12: \hspace{1em} \textbf{for} $j \leftarrow 1 : n$ \textbf{do}
13: \hspace{2em} $y^i_j = \{ u^i_j, \text{Ifrand} \leq 0.5 \}
14: \hspace{2em} x^i_j, \text{otherwise}$
15: \hspace{2em} \textbf{end for}
16: \hspace{1em} $F_C(i) = \{f(y^i_1), \ldots, f(y^i_N)\} \leftarrow \text{Evaluate}(\{y^i_1, \ldots, y^i_N\}^T)$
17: \hspace{1em} $G_C(i) = \{g(y^i_1), \ldots, g(y^i_N)\} \leftarrow \text{Evaluate}(\{y^i_1, \ldots, y^i_N\}^T)$
18: \hspace{1em} else
19: \hspace{2em} $v^i = \omega v^i + a_1 r^i (pbest^i - x^i) + a_2 r^i (nbest^i - x^i)$
20: \hspace{2em} $y^i = x^i + v^i$
21: \hspace{2em} \textbf{Apply the penalty} % (i.e., For Static Penalty Function referred to algorithm (2), or for Adaptive Penalty Function referred to algorithm (3)).
22: \hspace{2em} if $G(y^i) = 0$ then
23: \hspace{3em} if $f(y^i) < f(x^i)$ then
24: \hspace{4em} $x^i = y^i$
25: \hspace{3em} else
26: \hspace{4em} $x^i = x^i$
27: \hspace{3em} \textbf{end if}
28: \hspace{2em} \textbf{end if}
29: \hspace{2em} if $G(y^i) \neq 0$ then
30: \hspace{3em} $v(y^i) < v(x^i)$
31: \hspace{4em} $y^i = y^i$
32: \hspace{3em} else
33: \hspace{4em} $v(x^i) < v(y^i)$
34: \hspace{4em} $y^i = x^i$
35: \hspace{3em} \textbf{end if}
36: \hspace{2em} \textbf{end if}
37: \hspace{1em} \textbf{end if}
38: \hspace{1em} \textbf{end for}
39: \textbf{end for}

while developing the constrained EAs. The algorithm (2) explains the procedure of the suggested static penalty function.

2.2. Adaptive Penalty Functions. Static penalty functions are adjusted based on error-trial procedure. They are characterised by repeated, varied and continued attempts until success not achieved [61]. In general, these strategies are problem-dependent and the users have facing difficulties to settle down the parameters involved at different levels of constraints violation. This tedious and difficult task can overcome with the strategy
Algorithm 2 Procedure for Penalty Function in HCEA with Static Strategy

1: Input=N, v
2: Output=\text{f}(x)
3: \text{N}: Number of constraints;
4: v: Constrained function value;
5: l: Number of violation level;
6: \text{f}(x): Penalized Constrained function value;
7: \text{v}_m = \text{mean}(v)
8: for i \leftarrow 1 : N do
9: \text{v}_{\text{max}} = \text{max}(v);
10: \text{v}_1 = \text{if}(v > 0) \ & \ (v \leq 0.1 * \text{v}_m)
11: \text{R}_1 = 0.1 * \text{v}_m
12: \text{v}_2 = \text{if}(v > 0.1 * \text{v}_m) \ & \ (v \leq 0.2 * \text{v}_m)
13: \text{R}_2 = 0.2 * \text{v}_m
14: \text{v}_3 = \text{if}(v > 0.2 * \text{v}_m) \ & \ (v \leq \text{v}_m)
15: \text{R}_3 = \text{v}_m
16: \text{v}_4 = \text{if}(v > \text{v}_m) \ & \ (v \leq \text{v}_{\text{max}})
17: \text{R}_4 = \text{v}_{\text{max}}
18: end for
19: Return(\text{f}(x));

of adaptive penalty functions procedures [22, 9, 36]. The adaptive approaches utilize previous information in order to adjust the coefficient of the penalty functions. The suggested algorithm employ adaptively the following Penalty functions.

\begin{equation}
F(x) = d(x) + p(x)
\end{equation}

Where \( p(x) \) is the penalty value. In equation (2.2), the distance value \( d(x) \) is computed as follow:

\begin{equation}
d(x) = \begin{cases} 
\nu(x), & \text{if } r_f = 0 \\
\sqrt{\frac{f(x)^{\prime \prime 2}}{f(x)^{\prime \prime 2}}} + \nu(x)^2, & \text{otherwise}
\end{cases}
\end{equation}

\begin{equation}
r_f = \frac{\text{Number of feasible solution}}{\text{population size}}
\end{equation}

Where \( \nu(x) \) is the overall constrain violation.

\begin{equation}
f(x)^{\prime \prime} = \frac{f(x) - f_{\text{min}}}{f_{\text{max}} - f_{\text{min}}}
\end{equation}

Where \( f_{\text{max}} \) and \( f_{\text{min}} \) are maximum and minimum value of objective function. The penalty value is defined as follow

\begin{equation}
p(x) = (1 - r_f)M(x) + r_fN(x)
\end{equation}

Where \( M(x) \) and \( N(x) \) are given by

\begin{equation}
M(x) = \begin{cases} 
0, & \text{if } r_f = 0 \\
\nu(x), & \text{otherwise}
\end{cases}
\end{equation}
Algorithm 3 Procedure for Adaptive Penalty Function in the Framework HCEA

1: Input=F,G,N
2: Output=f_2
3: F: Fitness function value;
4: G: Constrained function value;
5: N: Population size;
6: x_f: Number of feasible solution;
7: f_2: Penalized Constrained function value;
8: G = (G > 0) * G
9: g_{max} = max(G)
10: w = find(g_{max} \neq 0)
11: if w = \phi then
12: v = 0
13: else
14: \mu(x) = \frac{\sum_{i=1}^{m} w_i(G_i(x))}{\sum_{i=1}^{m} w_i}
15: end if
16: if f_{max} = f_{min} then
17: f(x)'' = 1
18: else
19: f(x)'' = \frac{f(x)-f_{min}}{f_{max}-f_{min}}
20: end if
21: x_f = find(v = 0)
22: r_f = \frac{x_f}{X}
23: if r_f = 0 then
24: X = 0
25: d = v
26: else
27: X = v
28: d = \sqrt{(f(x)''^2 + v^2)}
29: end if
30: Y = f(x)''
31: Y(x_f) = x_f
32: p = (1 - r_f) * X + (r_f * Y)
33: f_2 = d + p

3. Discussion on Experimental Results

All experiments were carried out in the following platform and parameter settings:

- Operating system: Windows XP Professional;
- Programming language of the algorithms: Matlab;
- CPU: Core i3 Quad 1.8 GHz;
- RAM: 4 GB DDR2 500 GB;
- Execution: 25 times each algorithm with different random seeds.
The maximum number of function evaluations provides the features of the used

The Parameters of PSO were settled as

The tolerance value

The maximum of generations is set to

Differential Evolution (DE) has used with

The sized of population,

The tolerance value

Experiments were conducted with following parameter settings.

- The sized of population, \( N = 60 \);
- The tolerance value \( \epsilon \) for the equality constraints is set to 0.0001
- The Parameters of PSO were settled as \( \omega = 1 \);
- Differential Evolution (DE) has used with \( F = 0.7 \) and \( CR = 1.0 \);
- The maximum number of function evaluations 300,000;
- The tolerance value \( \Delta = 0.0001 \) for the problems consisting equality constraints.
- The maximum of generations is set to 2500.

Due largely to the nature of evolutionary algorithms (EAs), their behaviors and performances are mainly experimentally analyzed over different kinds of test suites of optimization and search problems. Several continuous test functions are already proposed for EC community over the last few years. These test functions played crucial role in developing and in studying the algorithmic behavior of particular evolutionary algorithm.

In this paper, we have used 24 benchmark functions that were designed for the special session of the IEEE Congress of Evolutionary Computation (CEC’2006). This CEC’2006 test suit consist of 24 benchmark functions comprising different characteristics like linear, nonlinear, polynomial, quadratic and cubic of objective functions with high dimensionality and wide range of linear inequalities (LI), nonlinear inequalities (NI), linear equalities (LE), and nonlinear equalities (NE) and number of other constraints \[28\]. The characteristics of the used Benchmark Functions are summarized in the Table 1.

Table 1. Classification and Properties of the used Benchmark Functions \[28\].

<table>
<thead>
<tr>
<th>Problem</th>
<th>( n )</th>
<th>Type of Function</th>
<th>( \rho )</th>
<th>LI</th>
<th>NI</th>
<th>LE</th>
<th>NE</th>
<th>( \alpha )</th>
<th>Known Optimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>g01</td>
<td>13</td>
<td>Quadratic</td>
<td>0.0111%</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>-15,000000000000</td>
</tr>
<tr>
<td>g02</td>
<td>20</td>
<td>Nonlinear</td>
<td>99.9971%</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-0.830319000000</td>
</tr>
<tr>
<td>g03</td>
<td>10</td>
<td>Polynomial</td>
<td>0.0000%</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>-1.000000000000</td>
</tr>
<tr>
<td>g04</td>
<td>5</td>
<td>Quadratic</td>
<td>32.1230%</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>-30665.35900000000</td>
</tr>
<tr>
<td>g05</td>
<td>4</td>
<td>Cubic</td>
<td>0.0000%</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>5126.498100000000</td>
</tr>
<tr>
<td>g06</td>
<td>2</td>
<td>Cubic</td>
<td>0.0000%</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>-6961.813800000000</td>
</tr>
<tr>
<td>g07</td>
<td>10</td>
<td>Quadratic</td>
<td>0.0000%</td>
<td>3</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>24.360900000000</td>
</tr>
<tr>
<td>g08</td>
<td>2</td>
<td>Nonlinear</td>
<td>0.8560%</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-0.095825000000</td>
</tr>
<tr>
<td>g09</td>
<td>7</td>
<td>Polynomial</td>
<td>0.5121%</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>680.6300573000000</td>
</tr>
<tr>
<td>g10</td>
<td>8</td>
<td>Linear</td>
<td>0.0010%</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>7049.248000000000</td>
</tr>
<tr>
<td>g11</td>
<td>2</td>
<td>Quadratic</td>
<td>0.0000%</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0.756000000000</td>
</tr>
<tr>
<td>g12</td>
<td>3</td>
<td>Quadratic</td>
<td>4.7713%</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1.000000000000</td>
</tr>
<tr>
<td>g13</td>
<td>5</td>
<td>Nonlinear</td>
<td>0.0000%</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>0.553949800000</td>
</tr>
<tr>
<td>g14</td>
<td>10</td>
<td>Nonlinear</td>
<td>0.0000%</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>-47.764888459500</td>
</tr>
<tr>
<td>g15</td>
<td>3</td>
<td>Quadratic</td>
<td>0.0000%</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>961.715022809000</td>
</tr>
<tr>
<td>g16</td>
<td>5</td>
<td>Nonlinear</td>
<td>0.0204%</td>
<td>4</td>
<td>34</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>1.905155258600</td>
</tr>
<tr>
<td>g17</td>
<td>6</td>
<td>Nonlinear</td>
<td>0.0000%</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>8853.533874806501</td>
</tr>
<tr>
<td>g18</td>
<td>9</td>
<td>Quadratic</td>
<td>0.0000%</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>-0.866025430800</td>
</tr>
<tr>
<td>g19</td>
<td>15</td>
<td>Nonlinear</td>
<td>33.4761%</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32.655592950200</td>
</tr>
<tr>
<td>g20</td>
<td>24</td>
<td>Linear</td>
<td>0.0000%</td>
<td>0</td>
<td>6</td>
<td>2</td>
<td>12</td>
<td>16</td>
<td>0.2409794002</td>
</tr>
<tr>
<td>g21</td>
<td>7</td>
<td>Linear</td>
<td>0.0000%</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>5</td>
<td>6</td>
<td>193.722451070000</td>
</tr>
<tr>
<td>g22</td>
<td>22</td>
<td>Linear</td>
<td>0.0000%</td>
<td>0</td>
<td>1</td>
<td>8</td>
<td>11</td>
<td>19</td>
<td>236.4309750400</td>
</tr>
<tr>
<td>g23</td>
<td>9</td>
<td>Linear</td>
<td>0.0000%</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>-400.035410000000</td>
</tr>
<tr>
<td>g24</td>
<td>2</td>
<td>Linear</td>
<td>79.6566%</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>-5.580913716000</td>
</tr>
</tbody>
</table>

The tolerance value \( \epsilon \) for the equality constraints is set to 0.0001.
Table 2. Comparative Analysis of the a) HEA-static and b) HEA-adaptive versus c) CJADE-D [53].

<table>
<thead>
<tr>
<th>Problem</th>
<th>Method</th>
<th>Minimum</th>
<th>Mean</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>g01</td>
<td>a</td>
<td>-15.000000000000</td>
<td>-15.000000000000</td>
<td>-15.000000000000</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>-14.999843000000</td>
<td>-14.999843000000</td>
<td>-14.999843000000</td>
</tr>
<tr>
<td></td>
<td>c</td>
<td>-14.999339</td>
<td>-12.969465</td>
<td>-14.664120</td>
</tr>
<tr>
<td>g02</td>
<td>a</td>
<td>-0.803619000000</td>
<td>-0.803619000000</td>
<td>-0.803619000000</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>-0.783840000000</td>
<td>-0.783840000000</td>
<td>-0.782220000000</td>
</tr>
<tr>
<td></td>
<td>c</td>
<td>-0.802628</td>
<td>-0.801371</td>
<td>-0.801388</td>
</tr>
<tr>
<td>g03</td>
<td>a</td>
<td>-1.000500000000</td>
<td>-1.000500000000</td>
<td>-1.000500000000</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>-0.562186000000</td>
<td>-0.562186000000</td>
<td>-0.562186000000</td>
</tr>
<tr>
<td></td>
<td>c</td>
<td>-99675.835134</td>
<td>-99689.240788</td>
<td>-99690.148408</td>
</tr>
<tr>
<td>g04</td>
<td>a</td>
<td>30665.538671999999</td>
<td>30665.5386720000162</td>
<td>30665.538671999999</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>30664.917622000001</td>
<td>30664.917621999986</td>
<td>30664.917622000001</td>
</tr>
<tr>
<td></td>
<td>c</td>
<td>-32196.152588</td>
<td>-32192.275678</td>
<td>-32192.360668</td>
</tr>
<tr>
<td>g05</td>
<td>a</td>
<td>6961.813876000000</td>
<td>6961.813875999722</td>
<td>6961.813876000000</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>6961.781377000000</td>
<td>6961.781377000246</td>
<td>6961.781377000000</td>
</tr>
<tr>
<td></td>
<td>c</td>
<td>7943.279110</td>
<td>7943.279110</td>
<td>7943.279110</td>
</tr>
<tr>
<td>g07</td>
<td>a</td>
<td>24.306209000000</td>
<td>24.306209000000</td>
<td>24.306209000000</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>25.857273000000</td>
<td>25.857273000000</td>
<td>25.857273000000</td>
</tr>
<tr>
<td></td>
<td>c</td>
<td>24.321855</td>
<td>24.321855</td>
<td>24.306209</td>
</tr>
</tbody>
</table>

equality constraints and NE is the number of nonlinear equality constraints and $a$ is the number of active constraints [28].

Tables 2-3-4 provide the experimental results of the suggested algorithms, namely, HCEA-static and HCEA-adaptive in comparison with recently developed constrained version of JADE aestivated as CJADE [53]. Tables 2-3-4 clearly show that the minimum functions values are much closer to known global optimal values of the test problems. Furthermore, Table 2-3-4 clearly indicated that the suggested HEA-static has found promising results in terms of better convergence toward the know optimal values while solving the g01, g02, g13, g18 and g21 problems. Similarly HCEA-static has tackled the problems g01, g02, g07, g13, g18, g19 and g21 with better mean functions values. The same is the case with median values for g01, g02, g13, g18 and g21 problems. It is also important to noted here that HCEA-static has performed better than our HCEA-adaptive and existing sate-of-the-art CJADE. The better performance could be attributed to better choice of efficient penalty functions and the combined use of DE [52] and PSO [25] in the framework of the proposed algorithm.

3.1. Graphical Result of HCEA-static. Figures 1-2-3 demonstrate the graphical results displayed by HCEA-static penalty functions for CEC’2006 benchmark functions. The figure 1 represents the evolution in minimum function values of the benchmark functions g1,g2,g3,g4, g6 and g7 provided by HCEA-static algorithm in 25 independent runs of simulations with different random seeds.Similiarly, figure 2 shows the convergence behaviour of the HCEA-static over g8,g9,g10,g11,g12 and g14 benchmark functions. Figure 3 display the convergence speed of the suggested HEA-static over g15,g16,g18,g19,g23 and g24 of the IEEE-CEC06 benchmark functions.
Figure 1. Convergence Graph of HCEA-static for CEC’06 Benchmark Functions.
Figure 2. Convergence Graph of HCEA-static for the CEC’06 Benchmark Functions.
Figure 3. Convergence Graph of HCEA-static for the CEC’06 Benchmark Functions.
Figure 4. Convergence Graph of HCEA-adaptive for the CEC’06 Benchmark Functions.
Figure 5. Convergence Graph of HCEA-adaptive for the CEC’06 Benchmark Functions.
Figure 6. Convergence Graph of the HCEA-adaptive for the CEC’06 Benchmark Functions.
Table 3. Comparative Analysis of the a) HEA-static, b) HEA-adaptive and c) CJADE-D [53].

<table>
<thead>
<tr>
<th>Problem</th>
<th>Method</th>
<th>Minimum</th>
<th>Mean</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>g08</td>
<td>a</td>
<td>−0.095825000000</td>
<td>−0.095825000000</td>
<td>−0.095825000000</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>−0.095825000000</td>
<td>−0.095825000000</td>
<td>−0.095825000000</td>
</tr>
<tr>
<td></td>
<td>c</td>
<td>−1296.8086712</td>
<td>−1348.735625</td>
<td>−1346.209597</td>
</tr>
<tr>
<td>g09</td>
<td>a</td>
<td>680.630057000000</td>
<td>680.630057000000</td>
<td>680.630057000000</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>680.813616000000</td>
<td>680.813616000044</td>
<td>680.813616000000</td>
</tr>
<tr>
<td></td>
<td>c</td>
<td>680.630057</td>
<td>680.630057</td>
<td>680.630057</td>
</tr>
<tr>
<td>g10</td>
<td>a</td>
<td>7049.248021000000</td>
<td>7049.248020999572</td>
<td>7049.248021000000</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>7362.948704000000</td>
<td>7362.948704000238</td>
<td>7362.948704000000</td>
</tr>
<tr>
<td></td>
<td>c</td>
<td>2131.846129</td>
<td>2140.683593</td>
<td>2140.014032</td>
</tr>
<tr>
<td>g11</td>
<td>a</td>
<td>0.749900000000</td>
<td>0.749900000000</td>
<td>0.749900000000</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>0.749985000000</td>
<td>0.749985000000</td>
<td>0.749999000000</td>
</tr>
<tr>
<td></td>
<td>c</td>
<td>0.443553</td>
<td>1.834e-003</td>
<td>0.440157</td>
</tr>
<tr>
<td>g12</td>
<td>a</td>
<td>−1.00000000000000</td>
<td>−1.0000000000000</td>
<td>−1.0000000000000</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>−1.00000000000000</td>
<td>−1.0000000000000</td>
<td>−1.0000000000000</td>
</tr>
<tr>
<td></td>
<td>c</td>
<td>−1.000000</td>
<td>−1.000000</td>
<td>−1.000000</td>
</tr>
<tr>
<td>g13</td>
<td>a</td>
<td>0.062460000000</td>
<td>0.075469168400</td>
<td>0.484833000000</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>0.086195000000</td>
<td>0.086195000000</td>
<td>0.086195000000</td>
</tr>
<tr>
<td></td>
<td>c</td>
<td>1.201125</td>
<td>1.837891</td>
<td>1.77548</td>
</tr>
<tr>
<td>g14</td>
<td>a</td>
<td>−47.764888000000</td>
<td>−47.764888000002</td>
<td>−47.764888000000</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>−44.836489900000</td>
<td>−44.83648999998</td>
<td>−44.836489000000</td>
</tr>
<tr>
<td></td>
<td>c</td>
<td>−1420.022548</td>
<td>−1561.624708</td>
<td>−1570.929559</td>
</tr>
<tr>
<td>g15</td>
<td>a</td>
<td>961.715022000000</td>
<td>961.715022000016</td>
<td>961.715022000000</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>961.718600000000</td>
<td>961.718599999989</td>
<td>961.721756000000</td>
</tr>
<tr>
<td></td>
<td>c</td>
<td>771.641627</td>
<td>709.556725</td>
<td>709.110988</td>
</tr>
</tbody>
</table>

3.2. Graphical Result of HCEA-adaptive for Benchmark Functions. Figures 1-2-3) demonstrate the convergence graph of the each CEC’06 test function displayed by HEA-static algorithm in single run of simulation. These figures clearly demonstrate that HEA-static have obtained approximate optimal solutions for g01, g04, g05, g07, g08, g09, g10, g12, g14, g15, g16, g19 and g24 in almost 500 generations. For the test problems denoted by g02, g03 and g18, HEA-static have obtained optimal solution in almost 1000 generations. These sort of convergence behaviors of the HEA-static stamped their fast convergence speed.

The Figures 4-5-6) depicts the convergence graph of the CPSO-Adaptive. These figures show that CPSO-Adaptive have figured out the problem g06, g07, g08, g10, g11, g12, g16, g19 and g24 in 500 generations to reach near the know optimal values of these problems. Similarly, the approximated optimal of problems g01, g02, g04, g15 and g18 are hereby obtained by HEA-Adaptive in almost in 1000 generations while for the g03, g05, g09 g13 and g17 the optimal values are almost obtained 2000 generations.

From the above discussion, one can conclude that the convergence behavior of the HCEA-static is much better than HEA-adaptive for the most of the used test problems. This better performance of the HCEA-static can be attributed to the fact to wise adjustment of the intrinsic parameters of the HCEA-static keeping in view the mathematical formulations demand of the IEEE-CEC’06 test problems [31].
<table>
<thead>
<tr>
<th>Problem</th>
<th>Method</th>
<th>Minimum</th>
<th>Mean</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>g16</td>
<td>a</td>
<td>-1.905155000000</td>
<td>-1.905155000000</td>
<td>-1.905155000000</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>-1.904886000000</td>
<td>-1.904886000000</td>
<td>-1.904886000000</td>
</tr>
<tr>
<td></td>
<td>c</td>
<td>-1.905155</td>
<td>-1.905155</td>
<td>-1.905155</td>
</tr>
<tr>
<td>g17</td>
<td>a</td>
<td>8862.697056999999</td>
<td>8862.697056999999</td>
<td>8862.697056999999</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>8875.006079999999</td>
<td>8876.330849559627</td>
<td>8904.658104000000</td>
</tr>
<tr>
<td></td>
<td>c</td>
<td>453.620550</td>
<td>1475.586258</td>
<td>1506.524787</td>
</tr>
<tr>
<td>g18</td>
<td>a</td>
<td>-0.866025000000</td>
<td>-0.866025000000</td>
<td>-0.866025000000</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>-0.756597000000</td>
<td>-0.753730160400</td>
<td>-0.747860000000</td>
</tr>
<tr>
<td></td>
<td>c</td>
<td>32.185020</td>
<td>87.105215</td>
<td>90.087847</td>
</tr>
<tr>
<td>g19</td>
<td>a</td>
<td>32.655593000000</td>
<td>32.655593000000</td>
<td>32.655594000000</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>41.652554000000</td>
<td>41.652554000000</td>
<td>41.652554000000</td>
</tr>
<tr>
<td></td>
<td>c</td>
<td>33.110947</td>
<td>32.655593</td>
<td>32.655593</td>
</tr>
<tr>
<td>g21</td>
<td>a</td>
<td>193.724510000000</td>
<td>193.724509999998</td>
<td>193.724510000000</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>194.219392000000</td>
<td>194.219392000000</td>
<td>194.219392000000</td>
</tr>
<tr>
<td></td>
<td>c</td>
<td>475.884809</td>
<td>345.569134</td>
<td>345.311741</td>
</tr>
<tr>
<td>g23</td>
<td>a</td>
<td>-400.055106000000</td>
<td>-400.055099999998</td>
<td>-400.055100000000</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>-373.588630000000</td>
<td>-373.588630000011</td>
<td>-373.588630000000</td>
</tr>
<tr>
<td></td>
<td>c</td>
<td>-3022.665686</td>
<td>-2458.257132</td>
<td>-2439.892090</td>
</tr>
<tr>
<td>g24</td>
<td>a</td>
<td>-5.508013000000</td>
<td>-5.508013000000</td>
<td>-5.508013000000</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>-5.507999000000</td>
<td>-5.507999000000</td>
<td>-5.507999000000</td>
</tr>
<tr>
<td></td>
<td>c</td>
<td>-5.508013</td>
<td>-5.628632</td>
<td>-5.631511</td>
</tr>
</tbody>
</table>

**4. Conclusion**

In general, classical optimization methods are usually unable to solve the problems having complicated objective functions and concave feasible regions with very small part of the whole search space. In the recent few years evolutionary algorithms (EAs) have become a research interest to different domain of researchers for solving complex problems in science, engineering, management and financial real applications due to their population-based nature. They don’t demand for any derivative information regarding the problems at hand. They provide a set of optimal solutions in single simulation unlike traditional optimization techniques.

Over the last two decades, several bio-inspired techniques have been developed based on nature collection, intelligence movement, thinking behaviors of social insects such as Ant, Honey Bees, Buffalos, Birds, Particles, Fishes etc. Particle swarm optimization (PSO) and differential evolution are two well known and the most effective EAs for engineering optimization problems. In this paper, we have combined both PSO and DE by employing two penalty functions with static manner and adaptive manner. The suggested algorithms have two versions called HCEA-adaptive and HCEA-static. The suggested algorithms have tackled most of the benchmark functions that were designed in 2006 IEEE-Congress on evolutionary computation (CEC’06). The simulation results offered by proposed algorithms are highly promising. Out of 24 benchmark function, 22 functions are solved by the suggested hybrid constrained EAs with good convergence speed as compared to the recent constrained version of JADE [53].
In near future, we intend to improve further the algorithmic structure of the suggested algorithms by employing some other novel and specialized constraint-handling techniques to cope with IEEE-CEC test instances [29, 10, 30].

Acknowledgements

The first author is thankful to Higher Education Commission Pakistan for the financial support under NRPU research project NO: 5892 to produce and publish this research article. The first author is also thankful to the Hybrid Metaheuristic Computing Research Group of King Khalid University (KKU), Kingdom of Saudi Arabia for the partial support to make it possible this research contribution.

References


Shah, Habib, Tairan, Nasser, Ghazali, Rozaida, Yeniay, Ozgur and Mashwani, Wali Khan, \textit{Hybrid Honey Bees Meta-Heuristic For Benchmark Data Classification}, Exploring Critical Approaches of Evolutionary Computation, IGI Global Publisher, 2019


