Improvised Scout Bee Movements in Artificial Bee Colony

Tarun Kumar Sharma
Amity Institute of Information Technology, Amity University Rajasthan, India
E-mail: taruniitr1@gmail.com

Millie Pant
Department of Applied Science and Engineering, IIT Roorkee, India
E-mail: millidma@gmail.com

Abstract — In the basic Artificial Bee Colony (ABC) algorithm, if the fitness value associated with a food source is not improved for a certain number of specified trials then the corresponding bee becomes a scout to which a random value is assigned for finding the new food source. Basically, it is a mechanism of pulling out the candidate solution which may be entrapped in some local optimizer due to which its value is not improving. In the present study, we propose two new mechanisms for the movements of scout bees. In the first method, the scout bee follows a non-linear interpolated path while in the second one, scout bee follows Gaussian movement. Numerical results and statistical analysis of benchmark unconstrained, constrained and real life engineering design problems indicate that the proposed modifications enhance the performance of ABC.

Index Terms — ABC, Artificial Bee Colony, Quadratic Interpolation, Gaussian distribution.

I. INTRODUCTION

The past few decades several nature inspired algorithms (NIA) have emerged as a potential tool for solving global optimization problems. Global optimization is an active area of research as most of the real life problems occurring in diverse areas can be modeled as optimization problems. More often, it is not only desirable but also necessary to obtain the global optimum rather than a local optimum. We may categorize the global optimization methods as traditional and nontraditional. Traditional methods include the classical techniques like gradient based methods (steepest descent, Newton and quasi Newton methods etc) while the nontraditional ones include Genetic Algorithms, Ta-boo Search, Simulated Annealing methods. Basically the traditional methods depend largely on the mathematical properties of the objective function and the search domain and therefore have a restricted application.

Nontraditional methods for global optimization have become more popular because of their generic nature which leads to wide applicability of these algorithms. Also it has been observed that these algorithms are also capable of locating the global optimum with a higher probability.

In the past few decades many nontraditional algorithms have been developed most of which are inspired by some natural phenomena.

The present study deals with artificial bee colony algorithm (ABC) a new computational technique proposed by Karaboga [1], based on the foraging behavior of honey bee swarm. ABC comes under the umbrella term of Swarm Intelligence (SI) algorithms which mimic the social behavior displayed by various species. Popular algorithms belonging to SI group are Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO) and ABC, the recent addition to SI group.

The performance of ABC is competitive to other population-based algorithms with an advantage of employing fewer control parameters [2 - 3], simplicity and ease of implementation. ABC has captured much attention from researchers and has been applied to solve many practical optimization problems [4 - 8], since its invention in 2005. However, similar to other nontraditional optimization methods, ABC also has some drawbacks which hamper its performance. For example, the convergence speed of ABC is typically slower than some other population-based algorithms like differential evolution (DE) [9 - 10] and PSO [11] when handling unimodal problems [6]. Also, ABC can easily get trapped in the local optima when solving complex multimodal problems [6]. It has also been observed that the search equation of ABC used to generate new candidate solutions, based on the information of previous solutions is good at exploration but is poor at exploitation [12], which results in an imbalance between exploration and exploitation. Therefore, accelerating convergence speed and avoiding the local optima have become two important and appealing goals in ABC research. A number of ABC variants have been proposed in literature to achieve these two goals [12 - 14].

ABC is based on the idea of division of labor in the colony of honey bees, (consisting of employed bees, onlooker bees and scout bees) for the search of potential food sources. In this paper a variation is made in the movement of scout bees. In the basic ABC algorithm,
scouts are assigned a random location for determining the new food location. This is generally done with the help of computer generated random numbers following uniform distribution, which may not prove to be very efficient for locating new food sources. In the present study we focus on enhancing the movement of scout bees in order to get more efficient food locations. Two enhancements are proposed in the scout bee phase (1) quadratic interpolation and (2) Gaussian movements. The first variant is named as QABC, while the second variant is named as GABC.

Here, we would like to mention that a part of this work has been published in a conference proceeding [15] but in the present paper we have sufficiently extended it by incorporating constrained and real life problems and have done a detailed analysis of the work.

The rest of the paper is organized as follows: section 2 gives an overview of ABC algorithm. In section 3, the proposed QABC and GABC are described. In section 4, experimental settings, evaluation criteria and results are given. Finally the paper concludes with section 5.

II. SURVEY OF LITERATURE

2.1 Unconstrained ABC

ABC classifies the foraging artificial bees into three groups, namely, employed bees, onlooker bees and scout bees. Half of the colony consists of employed bees, and the other half includes onlooker bees. In the foraging process of honeybee colonies, initially, some bees search randomly for food in a given area around the hive. After finding a food source, these bees take some nectar back to the hive, deposit the nectar and share the nectar information of the food sources with other bees waiting at the dance area (where waggle dance, Figure 1, is performed) within the hive.

The bee colony then enters a new cycle of iterations. At each iteration, following steps take place:

1. After sharing the information, an employed bee will either become an onlooker after abandoning the food source or continue to forage its previously visited site;
2. Some onlookers in the hive will simultaneously follow some employed bees based on the received information in order to further forage on some specific memorized food sources; and
3. Some scouts will spontaneously start a random search.

An important stage of the ABC algorithm, from which in fact the collective intelligence arises, is the sharing of information. This is achieved by influencing the behavior of onlookers which select their food source according to following probability $P_i$:

$$P_i = f_i / \sum_{k=1}^{SN} f_k$$  \hspace{1cm} (1)

Where $f_i$ is the fitness value of $i^{th}$ food source (position in parameter space). In other words onlookers will explore promising locations with higher probability than others. Candidate food sources are generated from memorized ones according to:

$$v_{ij} = x_{ij} + \phi_{ij}(x_{i,j} - x_{k,j})$$  \hspace{1cm} (2)

where $i,j = 1,...,SN$, $j = 1,...,n$, and $v_{ij}$ is the new food source generated by using both, the current food source $x_i$ and a randomly chosen food source $x_k$ from the population and $-1 \leq \phi_{ij} \leq 1$ (generated randomly every time it is used) determines the step size of the movement. Both, $i$ and $j$ are generated randomly such that $k \neq i$. When a source does not improve after a certain number of iterations, it is abandoned and replaced by the one found by a scout bee, using equation 2: which involves the generation of a new solution at random.

$$x_{i,j} = x_{\text{min},j} + \text{rand}(0,1)(x_{\text{max},j} - x_{\text{min},j})$$  \hspace{1cm} (3)

where $i = 1,2,...SN$, $j = 1,2,...,n$, $x_{\text{max},j}$ and $x_{\text{min},j}$ are upper and lower bounds of parameter $j$, respectively. These food sources are randomly assigned to $SN$ number of employed bees and their finesses are evaluated.

![Waggle dance of Honey bee in the hive](image)

**Figure 1:** Waggle dance of Honey bee in the hive

**Basic steps of Artificial Bee Colony:**

**Initialization of food sources (Population):** The initial population of solutions is filled with $SN$ number of randomly generated $n$-dimensional real-valued vectors (i.e., food sources). Let $X_i = \{x_{i,1}, x_{i,2},...,x_{i,n}\}$ represent the $i^{th}$ food source in the population, and then each food source is generated by equation (3).

**Employed bee initialization:** In this phase each employed bee $X_i$ generates a new food source $V_i$ in the neighborhood of its present position by using solution search equation (2). Once $V_i$ is obtained, it will be evaluated and compared to $X_i$. If the fitness of $V_i$ is equal to or better than that of $X_i$, $V_i$ will replace $X_i$ and become a new member of the population; otherwise $X_i$ is retained. In other words, a greedy selection mechanism is employed between the old and candidate solutions.

**Probabilistic Selection:** An important stage of the ABC algorithm, from which in fact the collective intelligence arises, is the sharing of information. This is
achieved by influencing the behavior of onlookers which will select their food source according to probability equation (1).

**Onlooker bee phase:** An onlooker bee evaluates the nectar information taken from all the employed bees and selects a food source \( X_i \) depending on its probability value \( P_i \). Once the onlooker has selected her food source \( X_i \), she produces a modification on \( X_i \) by using equation (2). As in the case of the employed bees, if the modified food source has a better or equal nectar amount than \( X_i \), the modified food source will replace \( X_i \) and become a new member in the population.

**Scout bee phase:** If a food source \( X_i \) cannot be further improved through a predetermined number of trials limit, the food source is assumed to be abandoned, and the corresponding employed bee becomes a scout. The scout produces a food source randomly using equation (3).

2.2 **Constrained ABC**

ABC algorithm for solving constrained optimization problems, we adopted the Pareto ranking method instead of the selection process (greedy selection) of the ABC algorithm described in the previous section. The following criteria are always enforced:

- Any feasible solution is preferred to any infeasible solution,
- Among two feasible solutions, the one having better objective function value is preferred,
- Among two infeasible solutions, the one having smaller constraint violation is preferred.

Because initialization with feasible solutions is very time consuming process and in some cases it is impossible to produce a feasible solution randomly, the ABC algorithm does not consider the initial population to be feasible. Structure of the algorithm already directs the solutions to feasible region in running process due to the selection process. Scout production process of the algorithm provides a diversity mechanism that allows new and probably infeasible individuals to be in the population. In order to produce a candidate food position from the old one in memory, the adapted ABC algorithm uses the following expression:

\[
V_i = \begin{cases} 
  x_{ij} + \phi_i (x_{ij} - x_{kj}), & \text{if } R_i \leq MR \\
  x_{ij}, & \text{otherwise}
\end{cases} \tag{4}
\]

where \( k \in \{1, 2, ..., SN\} \) is randomly chosen index. Although \( k \) is determined randomly, it has to be different from \( i \). \( R_i \) is randomly chosen real number in the range \([0,1]\) and \( j \in \{1, 2, ..., D\} \). MR, modification rate, is a control parameter that controls whether the parameter \( x_{ij} \) will be modified or not. In the version of the ABC algorithm proposed for constrained optimization problems, artificial scouts are produced at a predetermined period of cycles for discovering new food sources randomly. This period is another control parameter called scout production period (SPP) of the algorithm. At each SPP cycle, it is controlled if there is an abandoned food source or not. If there is, a scout production process is carried out.

**Detailed algorithm is discussed in Figure 2.**

**Algorithm: Constrained Artificial Bee Colony**

\[\begin{align*}
\text{Initialize } & \beta, \text{ the percentage of employed bees} \\
\text{Initialize } & n_{\text{max}} \text{ maximum number of explorer bees} \\
\text{Initialize } & \lambda, \text{ the foraging limit} \\
\text{Set } & n_s \text{ as the size of swarm} \\
\text{Set the fittest bee, } & \beta \text{ to null} \\
\text{Create and initialize to random positions } & n_e = n_p \text{ employed bees} \\
\text{Set the number of onlooker bees, } & n_o = n_s - n_e \\
\text{for each worker bee, } w_i \text{ do} \\
& \quad \text{Set } a_i = 0, \text{ where } a_i \text{ is the number of failed position updates} \\
& \quad \text{while stopping condition is false do} \\
& \quad \quad \text{Set } n_o = 0, \text{ where } n_o \text{ is the number of explorer bees} \\
& \quad \quad \text{for } i = 1, ..., n_o \text{ do} \\
& \quad \quad \quad \text{Randomly select } j \in [1, n_s], j \neq i \\
& \quad \quad \quad \text{Create a new position } v_{\text{new}} \text{ from worker bees using equation (4)} \\
& \quad \quad \quad \text{if } f(v_{\text{new}}) < f(x_{ai}) \text{ based on Pareto Ranking} \\
& \quad \quad \quad \text{Method then} \\
& \quad \quad \quad \quad x_{ai} = v_{\text{new}} \\
& \quad \quad \quad \text{end} \\
& \quad \quad \quad \text{else } \\
& \quad \quad \quad \quad a_i++ \\
& \quad \quad \quad \text{end} \\
& \quad \quad \quad \text{if } a_i > \lambda \text{ and } n_e < n_{\text{max}} \text{ then} \\
& \quad \quad \quad \quad n_e++ \\
& \quad \quad \quad \text{Move } w_i \text{ to a new random position in the search space} \\
& \quad \quad \quad a_i = 0 \\
& \quad \quad \text{end} \\
& \quad \text{end} \\
& \quad \text{for } i = 1, ..., n_o \text{ do} \\
& \quad \quad \text{Select } j \in [1, n_s] \text{ proportionate to } f(x_{ai}) \\
& \quad \quad \text{Set } o_i = w_j \\
& \quad \quad \text{Select } k \in [1, ..., n_s], k \neq j \\
& \quad \quad \text{Create a new position } v_{\text{new}} \text{ from onlooker bee } o_i \text{ and worker bee } w_j \text{ using equation (4)} \\
& \quad \quad \text{if } f(v_{\text{new}}) < f(x_{ai}) \text{ based on Pareto Ranking} \\
& \quad \quad \text{Method then} \\
& \quad \quad \quad x_{ai} = v_{\text{new}} \\
& \quad \quad \text{end} \\
& \quad \text{end} \\
& \quad \text{for each bee, } b_i \text{ in the swarm do} \\
& \quad \quad \text{if } f(x_{bi}) < f(x_{ai}) \text{ then} \\
& \quad \quad \text{end} \\
& \quad \quad \text{\textbf{end}} \\
& \quad \text{end} \\
& \text{Return } x_{ai} \text{ as Solution.}
\end{align*}\]

![Figure 2: Algorithm: Constrained Artificial Bee Colony](image)

### III. QABC AND GABC: THE PROPOSED VARIANTS

The proposed QABC and GABC differ from the basic ABC in terms of the movements of the scout bees.
Instead of assigning a random movement for the scout bees, in the proposed variants the scout bees are moved in order to find a better location for them. For this purpose we have used the method of interpolation and method following Gaussian distribution.

### Quadratic Interpolation

In the present study, the three initial solution vectors \( x_{i,j} \), \( x_{i,2} \), and \( x_{i,3} \) are selected randomly between 0 and 1, distinct from each other. From these three points, the coordinates of the new Food Location \( V_{i,G+1} = (v_{1,G+1}, v_{2,G+1}, \ldots, v_{n,G+1}) \) are determined as:

\[
\begin{align*}
    v_{1,i} &= (x_{1,i} - x_{2,i})^2 f(x_{1,i}) + (x_{2,i} - x_{1,i})^2 f(x_{2,i}) + (x_{1,i} - x_{2,i})^2 f(x_{3,i}) + (x_{2,i} - x_{3,i})^2 f(x_{1,i}) + (x_{3,i} - x_{1,i})^2 f(x_{2,i}) + (x_{3,i} - x_{2,i})^2 f(x_{3,i}) \\
    v_{2,i} &= (x_{2,i} - x_{3,i})^2 f(x_{2,i}) + (x_{3,i} - x_{2,i})^2 f(x_{3,i}) + (x_{2,i} - x_{3,i})^2 f(x_{1,i}) + (x_{3,i} - x_{1,i})^2 f(x_{2,i}) + (x_{1,i} - x_{3,i})^2 f(x_{1,i}) + (x_{1,i} - x_{2,i})^2 f(x_{3,i}) \\
    v_{3,i} &= (x_{3,i} - x_{1,i})^2 f(x_{3,i}) + (x_{1,i} - x_{3,i})^2 f(x_{1,i}) + (x_{3,i} - x_{1,i})^2 f(x_{2,i}) + (x_{1,i} - x_{2,i})^2 f(x_{3,i}) + (x_{2,i} - x_{1,i})^2 f(x_{2,i}) + (x_{2,i} - x_{3,i})^2 f(x_{1,i})
\end{align*}
\]

(4)

**Gaussian movement**: Gaussian distribution also called a “bell shaped curve” as a mode of perturbation (or mutation) has been used for generating new candidate solutions. The PDF for Gaussian function is given as: 

\[
f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} ; \text{ with mean 0 and standard deviation 1, i.e. } N(0, 1).
\]

**Proposed Variants**: After initialization QABC starts like the usual ABC as discussed in previous section and when the food source \( X_{i,j} \) cannot be further improved through a fixed trial limit, the food source get abandoned, and the corresponding employed bee act as scout bee. Than scout bee produces a new food source using, Gaussian random numbers and the equation is given as:

\[
x_{i,j} = x_{\text{min,j}} + \text{Gauss}(-1,1)\left( x_{\text{max,j}} - x_{\text{min,j}} \right)
\]

(5)

where, \( \text{Gauss} \) is a random number following Gaussian distribution. Fig. 3 explains the pseudocode of QABC and GABC & flow graph is shown in Fig. 4.

```
Begin
1. Initialize the population of food sources \( x_i, i = 1, \ldots, SN \)
2. Evaluate each food source \( x_i, i = 1, \ldots, SN \)
3. \( \text{cycle} = 1 \)
Repeat
For each food source \( x_i \) in the population
4. Generate a new food source \( v_i \) by its corresponding employed bee (equation (2))
5. Evaluate \( v_i \)
6. Keep the best solution between \( x_i \) and \( v_i \)
End
7. Select, based on fitness proportional selection, the food sources to be visited by onlooker bees
For each food source \( x_i \) chosen by an onlooker bee
8. Generate a new food source \( v_i \) by its corresponding onlooker bee (equation (2))
9. Evaluate \( v_i \)
10. Keep the best solution between \( x_i \) and \( v_i \)
End
11. Use the scout bee to replace those abandoned food sources using (equation (4) for QABC and equation (5) for GABC)
12. Keep the best solution between \( x_i \) and \( v_i \)
13. Save in memory the best food source so far
14. \( \text{cycle} = \text{cycle} + 1 \)
Until \( \text{cycle} \)
End
```

Figure 3: Pseudocode of Proposed QABC & GABC

![Flow Graph of Proposed Variants](Image)

Figure 4: Flow Graph of Proposed Variants
IV. EXPERIMENTAL SETTINGS, PERFORMANCE CRITERIA AND RESULTS

Benchmark and Engineering Problems: In order to assess the performance of the proposed QABC and GABC algorithm, we considered four unconstrained, six constrained and five engineering design given in Appendix I, II and III respectively.

Structure of the Problems: The general NLP is given by nonlinear objective function $f$, which is to be minimized/maximized with respect to the design variables $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and the nonlinear inequality and equality constraints. The mathematical models of the problems considered in the paper are of the type:

$$\begin{align*}
\text{Minimize}\ f(\mathbf{x}) \\
\text{Subject to:} & \quad g_j(\mathbf{x}) \leq 0, \quad j = 1, \ldots, p \\
& \quad h_k(\mathbf{x}) = 0, \quad k = 1, \ldots, q \\
& \quad x_{i_{\text{min}}} \leq x_i \leq x_{i_{\text{max}}} \quad (i = 1, \ldots, n)
\end{align*}$$

where $p$ and $q$ are the number of inequality and equality constraints respectively.

A set of six constrained benchmark problems taken from literature [16] is considered to evaluate the performance of the proposed QABC and GABC. All the problems are nonlinear in nature i.e. either the objective function or the constraints or both have a nonlinear term in it. The mathematical models of the problems along with the optimal solution are given in Appendix II.

Characteristics of test functions ($f$)

<table>
<thead>
<tr>
<th>$f$</th>
<th>n</th>
<th>Func.Type</th>
<th>$\rho$ (%)</th>
<th>li</th>
<th>ni</th>
<th>le</th>
<th>ne</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>g01</td>
<td>13</td>
<td>Quad.</td>
<td>0.0003</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>g02</td>
<td>20</td>
<td>Nonl.</td>
<td>99.9973</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>g03</td>
<td>10</td>
<td>Nonl.</td>
<td>0.0026</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>g04</td>
<td>5</td>
<td>Quad.</td>
<td>27.0079</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>g05</td>
<td>4</td>
<td>Nonl.</td>
<td>0.0000</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>g06</td>
<td>2</td>
<td>Nonl.</td>
<td>0.0057</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

Parameter $n$ denotes the number of parameters. The function can be linear, nonlinear (Nonl.) or quadratic (Quad.). $li$ is the number of linear inequality constraints, $ni$ is the number of nonlinear inequality constraints, $le$ is the number of linear equality constraints, $ne$ is the number of nonlinear inequality constraints, $a$ is the number of active restrictions and $\rho$ is a percentage of the feasible area. A percentage of feasible area is:

$$\rho = \frac{|F|}{|S|}$$

where $|F|$ is the number of feasible solutions and $|S|$ is the total number of solutions randomly generated. Michalewicz and Schoenauer [17] suggested a total number of 1,000,000 solutions for $|S|$.

Engineering Design Problems: The credibility of an optimization algorithm also depends on its ability to solve real life/engineering’s problems. In this paper we took five real life engineering design problems to validate the efficiency of all the proposed algorithms.

Mathematical models of problems are given in Appendix III.

Experimental Settings: The proposed algorithms are tested on 4 unconstrained, 6 constrained benchmark problems and 5 engineering problems given in Appendix I, II and III respectively. The following control parameters are: colony size (SN), MCN (Maximum Cycle Numbers) and “VTR”. ABC and the proposed variants are implemented on Dev-C++ and the experiments are conducted on a computer with 2.00 GHz Intel (R) core (TM) 2 duo CPU and 2- GB of RAM. For each problem, all the algorithms independently run 30 times. The parameter setting is taken as follows:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population size</td>
<td>100</td>
</tr>
<tr>
<td>SN</td>
<td>100</td>
</tr>
<tr>
<td>Value to Reach (VTR)</td>
<td>0.15</td>
</tr>
<tr>
<td>MCN</td>
<td>8000</td>
</tr>
</tbody>
</table>

Constrained and Engineering Design Problems

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Colony Size</td>
<td>20</td>
</tr>
<tr>
<td>Limit</td>
<td>MCN/(2*SN)</td>
</tr>
<tr>
<td>Max. Cycle Numbers (MCN)</td>
<td>6000</td>
</tr>
<tr>
<td>NFE (for Engg. desing. Prob)</td>
<td>5.0E+06</td>
</tr>
<tr>
<td>Max. Func. Evaluations (Max. FEs)</td>
<td>6.0E+05</td>
</tr>
<tr>
<td>MR (Modification Rate)</td>
<td>0.4</td>
</tr>
<tr>
<td>Runs</td>
<td>30</td>
</tr>
</tbody>
</table>

Performance Criteria: Mean Fitness, Standard Deviation, Best and Worst: The average of function fitness value that an algorithm can find, using predefined MCN, is recorded in each run and then average of the function fitness values are calculated. Also the average, standard deviation, best and worst of the fitness values are calculated.

$MCN$: The MCN (Maximum Cycle Number) is recorded when the VTR is reached before to reach maximum MCN, i.e. we set the termination criteria as $|f_{\text{optimal}} - f_{\text{global}}| \leq \text{VTR}$ and record MCN over 30 runs.

$\text{Acceleration rate (AR)}$ in %: This criterion is used to compare the convergence speeds between ABC, GABC and QABC. It is defined as follows:

$$AR = \frac{MCN_{\text{ABC,GABC}} - MCN_{\text{GABC}}}{MCN_{\text{ABC,GABC}}} \times 100$$

Comparison Criteria, Results and Discussion: Comparison Criteria: We used several criteria to measure the performance of the proposed QABC and GABC algorithm and to compare it with basic ABC. In Tables 1 – 2 we recorded the performance the proposed QABC and GABC for unconstrained problems in terms
of best worst and average fitness function value along with the standard deviation (Std) while increasing the MCN to two different values 5000 with D=30 & SN=20 and 8000 with D=50 & SN=50. t-Test and acceleration rate is given in Table 3.

In Tables 4 and 5 the performance of QABC and GABC is compared with basic ABC for solving constrained optimization problems. The comparison criteria for all the algorithms taken in the present study are given as:

1. Feasible Run: A run during which at least one feasible solution is found in Max NFE.
2. Successful Run: A run during which the algorithm finds a feasible solution satisfying $(f(x) - f(x^*)) \leq 0.0001$.
3. Feasible Rate = (# of feasible runs) / total runs
4. Success Rate = (# of successful runs) / total runs
5. Success Performance = mean (FE’s for successful runs) * (# of total runs) / (# of successful runs)

Further, the average acceleration rate ($AR_{avg}$) and the average success rate ($SR_{avg}$) over test functions are calculated as follows:

$$AR_{avg} = \frac{1}{n} \sum_{i=1}^{n} AR_i$$

$$SR_{avg} = \frac{1}{n} \sum_{i=1}^{n} SR_i$$

In order to get a better insight into the relative performance of ABC, QABC, GABC, RABC, [18] and [19], the value of their performance is calculated. This performance gives specified importance to the number of function evaluations to observe the efficiency of the algorithm. For the computational algorithms under comparison the value of performance $P_j$ for the $j^{th}$ algorithm is computed as under:

$$P_j = \frac{1}{N} \sum_{i=1}^{N} (\alpha'_i)$$

where $\alpha'_i = \begin{cases} 
\frac{Mf_i}{Af}, & \text{if } Sr_i \geq 0, \\
0, & \text{if } Sr_i = 0
\end{cases}$

where $i = 1, 2, ..., N$.

Here, $Af^i$ is the average number of function evaluations used by the $j^{th}$ algorithm in obtaining the optimal solution of $i^{th}$ problem in the case of the successful runs, and $Mf^i$ the minimum of the average number of function evaluations of successful runs used of the algorithms under comparison in obtaining the optimal solution of $i^{th}$ problem. $N$ is total number of problems on which the performance of algorithms has been tested.

The larger the value of $P_j$ better is the performance of the algorithm.

For engineering design problems the performance of ABC and proposed variants is compared in terms of mean fitness value & standard deviation (Std.) given in Table 6 and in Table 7 the iterations and acceleration rate is given.

**Results and Discussions: Unconstrained Problems:**

In Table 2 we have taken the results on the basis of average error. In this case MCN is fixed at 5000 and 8000, to estimate the average of minimum fitness function value in 30 runs. From the Table 2 it can be clearly observed that for all benchmark functions QABC gives better results than GABC and ABC. A two tail sample $t$-test is also applied to analyze the statistical significance of the proposed algorithm. We have checked the significant difference of QABC with respect to GABC and ABC at 5% level of significance. The calculated $t$-value (Table 3) of all function is greater than $t$-table value that shows the significantly better performance in the comparison of ABC. In the Table-2, we fixed $VTR$ as given in experimental setting and then calculated the MCN of 30 runs. From Table-2 we can see that the proposed QABC gives the better results for every function in the comparison to the other algorithms. From the Table-3 it is clear that the proposed QABC is faster than GABC by 13.50% and ABC by 16.50%, when MCN=5000, D=30 & SN=20 and in other case when MCN=8000, D=50 & SN=50, QABC is again faster than GABC by12.27% and ABC by 10.57%. Best and worst function values in 30 runs are also presented in Table-1.

Fig. 4(a) & 4(b) shows the convergence graph Rosenbrock’s and Griewank function. MCN taken to estimate the average of minimum fitness function value in 30 runs are also presented graphically in Fig. 5.

**Constrained Problems:** The results obtained in Table 4 using QABC are better or equal to that of results obtained by original ABC, GABC algorithm and MO-ABC for constrained optimization problems. The g05 function illustrates that the QABC due to greater exploration capability, better best results are reached, but also the worst result is slightly worse than compared algorithm. The standard deviation for g02 function is also inferior for the same reason. QABC reaches much better results for g02 and g03 function then the original ABC algorithm.

The superior performance of QABC is more visible from Tables 4 and 5 where the results are recorded after fixing the accuracy at 0.0001. In these tables we can see that the proposed QABC gave a better or at par performance with the other three algorithms. We will now take the comparison criteria one-by-one and discuss them briefly. The first criterion is that of a feasible run. A run is said to be feasible if at least one feasible solution is obtained in maximum number of function evaluations. According to this criterion all the algorithm gave 100% feasible rate for all the test problems.
However, if we observe the second criterion which is of successful run and is recorded when the algorithm finds a feasible solution satisfying the given accuracy (ε=0.0001) it can be seen that the proposed QABC outperforms the other algorithms in all the test cases. The third criterion is that of failure performance which depends on the feasibility rate and success rate, as described in the previous subsection. Here also QABC gave a better performance in comparison to the other three algorithms taken for comparison.

Comparative acceleration rate is also shown graphically in Fig. 6 using radar representation, which explains that QABC is 4.18% better than ABC, 3.64% better than GABC, 58.11% better than [18] where as 68.7% better than [19] for g01. Similarly the graphs explain the acceleration rate for rest of the problems g02 - g06.

The performance comparison graph of algorithms is given in Fig. 7 (a). This curves (best solution versus NFE’s) show that QABC converges faster than others toward the optimal solution. Also Average Success performance and acceleration performance graph to solve six constrained benchmark problems are given in single graph shown in Fig. 7(b).

Engineering Design Problems: The results of engineering design problems are presented in Table 6 and 7. We see that in terms of average fitness function value and standard deviation all the algorithms gave more or less similar results although in some cases the proposed algorithms gave a marginally better performance than basic ABC and GABC. From Table 7 it can be clearly observed that QABC emerges clear winner in terms of iterations taken to reach the optimum value, further QABC is 24.36% faster than ABC where as 8.24% faster than GABC when compared in terms of acceleration rate.

<p>| TABLE 1: Mean, Standard Deviation, Best, Worst, Mean and values obtained by ABC, GABC and QABC through 30 independent runs on function from $f_1$ to $f_4$ |
|---|---|---|---|---|
| $F$ | Algorithm | $MCN=5000/D=30/SN=20$ | $MCN=8000/D=50/SN=50$ |</p>
<table>
<thead>
<tr>
<th></th>
<th>Mean Std. Dev</th>
<th>Best Worst</th>
<th>Mean Std. Dev</th>
<th>Best Worst</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1$</td>
<td>ABC</td>
<td>1.162e-015 1.896e-015</td>
<td>0.062e-015 2.235e-014</td>
<td>2.841e-014 3.867e-014</td>
</tr>
<tr>
<td></td>
<td>GABC</td>
<td>8.295e-016 2.531e-017</td>
<td>1.460e-016 3.310e-014</td>
<td>3.159e-015 9.268e-016</td>
</tr>
<tr>
<td></td>
<td>QABC</td>
<td>1.437e-016 4.448e-017</td>
<td>8.377e-016 4.626e-014</td>
<td>2.551e-014 0</td>
</tr>
<tr>
<td>$f_3$</td>
<td>ABC</td>
<td>2.016e-002 8.446e-003</td>
<td>1.953e-002 3.337e-002</td>
<td>1.075e-001 0</td>
</tr>
<tr>
<td></td>
<td>GABC</td>
<td>2.738e-004 8.152e-004</td>
<td>1.199e-004 7.972e-004</td>
<td>1.181e-002 0</td>
</tr>
<tr>
<td></td>
<td>QABC</td>
<td>2.264e-004 8.389e-005</td>
<td>9.163e-005 5.065e-001</td>
<td>9.753e-003 0</td>
</tr>
<tr>
<td>$f_4$</td>
<td>ABC</td>
<td>1.514e-016 4.309e-017</td>
<td>8.289e-017 1.514e-014</td>
<td>1.147e-015 3.265e-016</td>
</tr>
</tbody>
</table>
TABLE 2: MCN taken by functions for VTR (NC-Not Converge)

<table>
<thead>
<tr>
<th>F</th>
<th>D=30/SN=20</th>
<th>D=50/SN=50</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ABC</td>
<td>GABC</td>
</tr>
<tr>
<td>( f_1 )</td>
<td>1148</td>
<td>1102</td>
</tr>
<tr>
<td>( f_2 )</td>
<td>4011</td>
<td>3903</td>
</tr>
<tr>
<td>( f_3 )</td>
<td>NC</td>
<td>NC</td>
</tr>
<tr>
<td>( f_4 )</td>
<td>1083</td>
<td>1009</td>
</tr>
</tbody>
</table>

TABLE 3: T-test and AR(%), (here 3/1 implies QABC/ABC, and 3/2 implies QABC/GABC )

<table>
<thead>
<tr>
<th>F</th>
<th>3-test</th>
<th>AR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SN=20/D=30</td>
<td>SN=50/D=50</td>
</tr>
<tr>
<td>( f_1 )</td>
<td>1.611</td>
<td>40.20</td>
</tr>
<tr>
<td>( f_2 )</td>
<td>0.16</td>
<td>0.235</td>
</tr>
<tr>
<td>( f_3 )</td>
<td>7.080</td>
<td>0.173</td>
</tr>
<tr>
<td>( f_4 )</td>
<td>9.969</td>
<td>0.781</td>
</tr>
</tbody>
</table>

1.#NF indicates std. dev. = 0; NC Not Converge

Figure 5: Convergence plot of (a) Griekwank (b) Rosenbrock
TABLE 4: Simulation results for the constrained optimization problems

<table>
<thead>
<tr>
<th>$f$</th>
<th>Statistics</th>
<th>$g_{01}$</th>
<th>$g_{02}$</th>
<th>$g_{03}$</th>
<th>$g_{04}$</th>
<th>$g_{05}$</th>
<th>$g_{06}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ABC</td>
<td>MO-ABC</td>
<td>QABC</td>
<td>GABC</td>
<td>ABC</td>
<td>MO-ABC</td>
</tr>
<tr>
<td>SD</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.01E-14</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Best</td>
<td></td>
<td>-1</td>
<td>-1</td>
<td>-1.00037</td>
<td>-0.7563</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>Mean</td>
<td></td>
<td>-30665.539</td>
<td>-30665.539</td>
<td>-30665.531</td>
<td>-30665.539</td>
<td>-30665.539</td>
<td>-30665.539</td>
</tr>
<tr>
<td>SD</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Mean</td>
<td></td>
<td>5185.714</td>
<td>5162.496</td>
<td>5125.9462</td>
<td>5126.5067</td>
<td>5185.714</td>
<td>5162.496</td>
</tr>
<tr>
<td>SD</td>
<td></td>
<td>75.358</td>
<td>47.8</td>
<td>52.21</td>
<td>51.40</td>
<td>75.358</td>
<td>47.8</td>
</tr>
<tr>
<td>Worst</td>
<td></td>
<td>5438.387</td>
<td>5229.134</td>
<td>5232.179</td>
<td>5126.4967</td>
<td>5438.387</td>
<td>5229.134</td>
</tr>
<tr>
<td>Best</td>
<td></td>
<td>5126.484</td>
<td>5126.582</td>
<td>5126.531</td>
<td>5126.4967</td>
<td>5126.484</td>
<td>5126.582</td>
</tr>
<tr>
<td>SD</td>
<td></td>
<td>0.002</td>
<td>0</td>
<td>0.00001</td>
<td>1.98E-15</td>
<td>0.002</td>
<td>0</td>
</tr>
</tbody>
</table>
TABLE 5: Comparison Results: NFE (Best, Worst and Mean) to achieve the fixed accuracy level \((f(x) - f(x^*)) \leq 0.0001\), success rate, Feasible Rate and Success Performance for problems \(g01\) – \(g06\). Algo: Algorithms

<table>
<thead>
<tr>
<th>Problem</th>
<th>Algo</th>
<th>Best</th>
<th>Worst</th>
<th>Mean</th>
<th>Feasible Rate (%)</th>
<th>Success Rate (%)</th>
<th>Success Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>(g01)</td>
<td>ABC</td>
<td>25850</td>
<td>86500</td>
<td>33308</td>
<td>100</td>
<td>100</td>
<td>33308</td>
</tr>
<tr>
<td></td>
<td>QABC</td>
<td>25619</td>
<td>67997</td>
<td>31914</td>
<td>100</td>
<td>100</td>
<td>31914</td>
</tr>
<tr>
<td></td>
<td>GABC</td>
<td>27981</td>
<td>83536</td>
<td>33120</td>
<td>100</td>
<td>100</td>
<td>33120</td>
</tr>
<tr>
<td></td>
<td>[18]</td>
<td>25273</td>
<td>346801</td>
<td>76195</td>
<td>100</td>
<td>52</td>
<td>146528.8</td>
</tr>
<tr>
<td></td>
<td>[19]</td>
<td>95100</td>
<td>106900</td>
<td>101532</td>
<td>100</td>
<td>100</td>
<td>101532</td>
</tr>
<tr>
<td>(g02)</td>
<td>ABC</td>
<td>175832</td>
<td>428719</td>
<td>199273</td>
<td>100</td>
<td>100</td>
<td>199273</td>
</tr>
<tr>
<td></td>
<td>QABC</td>
<td>92764</td>
<td>192850</td>
<td>117290</td>
<td>100</td>
<td>100</td>
<td>117290</td>
</tr>
<tr>
<td></td>
<td>GABC</td>
<td>127823</td>
<td>215782</td>
<td>168382</td>
<td>100</td>
<td>100</td>
<td>168382</td>
</tr>
<tr>
<td></td>
<td>[18]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>100</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>[19]</td>
<td>180000</td>
<td>327900</td>
<td>231193</td>
<td>100</td>
<td>56</td>
<td>412844.6</td>
</tr>
<tr>
<td>(g03)</td>
<td>ABC</td>
<td>498266</td>
<td>587179</td>
<td>543319</td>
<td>100</td>
<td>58</td>
<td>936756.9</td>
</tr>
<tr>
<td></td>
<td>QABC</td>
<td>440982</td>
<td>479265</td>
<td>449071</td>
<td>100</td>
<td>76</td>
<td>590882.9</td>
</tr>
<tr>
<td></td>
<td>GABC</td>
<td>475387</td>
<td>511732</td>
<td>463196</td>
<td>100</td>
<td>59</td>
<td>785078</td>
</tr>
<tr>
<td></td>
<td>[18]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>100</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>[19]</td>
<td>450100</td>
<td>454000</td>
<td>450644</td>
<td>100</td>
<td>100</td>
<td>450644</td>
</tr>
<tr>
<td>(g04)</td>
<td>ABC</td>
<td>11956</td>
<td>13651</td>
<td>12651</td>
<td>100</td>
<td>100</td>
<td>12651</td>
</tr>
<tr>
<td></td>
<td>QABC</td>
<td>9659</td>
<td>10842</td>
<td>9915</td>
<td>100</td>
<td>100</td>
<td>9915</td>
</tr>
<tr>
<td></td>
<td>GABC</td>
<td>10750</td>
<td>12281</td>
<td>11917</td>
<td>100</td>
<td>100</td>
<td>11917</td>
</tr>
<tr>
<td></td>
<td>[18]</td>
<td>15363</td>
<td>25776</td>
<td>20546</td>
<td>100</td>
<td>100</td>
<td>20546</td>
</tr>
<tr>
<td></td>
<td>[19]</td>
<td>74300</td>
<td>85000</td>
<td>79876</td>
<td>100</td>
<td>100</td>
<td>79876</td>
</tr>
<tr>
<td>(g05)</td>
<td>ABC</td>
<td>21765</td>
<td>91985</td>
<td>25964</td>
<td>100</td>
<td>76</td>
<td>34163.16</td>
</tr>
<tr>
<td></td>
<td>QABC</td>
<td>13350</td>
<td>65400</td>
<td>19286</td>
<td>100</td>
<td>88</td>
<td>21915.91</td>
</tr>
<tr>
<td></td>
<td>GABC</td>
<td>19573</td>
<td>78194</td>
<td>57934</td>
<td>100</td>
<td>78</td>
<td>74274.36</td>
</tr>
<tr>
<td></td>
<td>[18]</td>
<td>94156</td>
<td>482411</td>
<td>364218</td>
<td>100</td>
<td>16</td>
<td>2276363</td>
</tr>
<tr>
<td></td>
<td>[19]</td>
<td>450100</td>
<td>457200</td>
<td>452256</td>
<td>100</td>
<td>100</td>
<td>452256</td>
</tr>
<tr>
<td>(g06)</td>
<td>ABC</td>
<td>9167</td>
<td>10284</td>
<td>9948</td>
<td>100</td>
<td>100</td>
<td>9948</td>
</tr>
<tr>
<td></td>
<td>QABC</td>
<td>7373</td>
<td>9892</td>
<td>8952</td>
<td>100</td>
<td>100</td>
<td>8952</td>
</tr>
<tr>
<td></td>
<td>GABC</td>
<td>8791</td>
<td>9972</td>
<td>9073</td>
<td>100</td>
<td>100</td>
<td>9073</td>
</tr>
<tr>
<td></td>
<td>[18]</td>
<td>16794</td>
<td>22274</td>
<td>20043</td>
<td>100</td>
<td>100</td>
<td>20043</td>
</tr>
<tr>
<td></td>
<td>[19]</td>
<td>47800</td>
<td>61100</td>
<td>56508</td>
<td>100</td>
<td>100</td>
<td>56508</td>
</tr>
</tbody>
</table>
Improvised Scout Bee Movements in Artificial Bee Colony

Figure 7: Radar representation of acceleration rate of constrained benchmark problems (a) ABC Vs. QABC (b) GABC Vs. QABC (c) [18] Vs. QABC (d) [19] Vs. QABC

Figure 8: (a) Average Success performance graph for all algorithms (b) Average Success performance and acceleration performance graph to solve six constrained benchmark problems.

TABLE 6: Mean of fitness function value and (standard deviation) and for all algorithms

<table>
<thead>
<tr>
<th>Function</th>
<th>ABC Mean (Std.)</th>
<th>GABC Mean (Std.)</th>
<th>QABC Mean (Std.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>169.849 (0.00801)</td>
<td>169.846 (1.7034e-016)</td>
<td>169.842 (1.6309e-016)</td>
</tr>
<tr>
<td>F2</td>
<td>4.21319 (3.1904e-016)</td>
<td>4.20916 (1.4291e-017)</td>
<td>4.20079 (.000101)</td>
</tr>
<tr>
<td>F3</td>
<td>4.7457e-008 (1.8741e-018)</td>
<td>1.8536e-009 8.0873e-023</td>
<td>2.9014e-009 (2.9443e-024)</td>
</tr>
<tr>
<td>F4</td>
<td>-26.0119 (0.71933)</td>
<td>-26.0317 (0.03981)</td>
<td>-26.0418 (0.65189)</td>
</tr>
<tr>
<td>F5</td>
<td>2.9975e+006 (0.22175)</td>
<td>2.9975e+006 (0.0)</td>
<td>2.7917e+006 (0.00319)</td>
</tr>
</tbody>
</table>

TABLE 7: Comparison of QABC with other algorithms in terms of Iterations and AR (%), Here 3/1 implies QABC vs. ABC and 3/2 implies QABC vs. GABC

<table>
<thead>
<tr>
<th>Function</th>
<th>ABC</th>
<th>GABC</th>
<th>QABC</th>
<th>AR 3/1</th>
<th>AR 3/2</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>289</td>
<td>249</td>
<td>197</td>
<td>31.83</td>
<td>20.88</td>
</tr>
<tr>
<td>F2</td>
<td>756</td>
<td>524</td>
<td>513</td>
<td>32.14</td>
<td>2.10</td>
</tr>
<tr>
<td>F3</td>
<td>418</td>
<td>317</td>
<td>321</td>
<td>23.21</td>
<td>---</td>
</tr>
<tr>
<td>F4</td>
<td>240000</td>
<td>240000</td>
<td>240000</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>F5</td>
<td>8517</td>
<td>6913</td>
<td>5568</td>
<td>34.62</td>
<td>19.46</td>
</tr>
<tr>
<td>Average Acceleration Rate (AR(%))</td>
<td>24.36</td>
<td>8.24</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
V. CONCLUSION

ABC is one of the latest additions to the class SI based algorithms. In the present study enhancements are suggested in the scout bee phase of the ABC algorithm. Instead of providing a random movement for scout bees, like the basic ABC algorithm, interpolation and Gaussian movements are embedded to generate more efficient food locations. The corresponding variants named as QABC and GABC are able to explore and exploit the search space more effectively. These variants are tested on 4 unconstrained, 6 constrained benchmark problems and five engineering design problems. Various performance criteria were considered to assess the proposed algorithms. Numerical results and graphical illustrations indicate that the proposed enhancements improve the performance of basic ABC in terms of convergence rate and fitness function.

REFERENCES

Appendix A

<table>
<thead>
<tr>
<th>Name of function</th>
<th>Definition</th>
<th>Range</th>
<th>Known global optimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Griewank</td>
<td>$f_1(x) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$</td>
<td>[-600, 6000]</td>
<td>$f_1(0,\ldots,0) = 0$.</td>
</tr>
<tr>
<td>Restrigin’s</td>
<td>$f_2 = 10n + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i))$</td>
<td>[-5.12, 5.12]</td>
<td>$f_2(0,\ldots,0) = 0$.</td>
</tr>
<tr>
<td>Rosenbrock’s</td>
<td>$f_3(x) = \sum_{i=1}^{n-1} \left[100(x_i^2 - x_{i+1}^2)^2 + (1 - x_{i+1})^2 \right]$</td>
<td>[-30, 30]</td>
<td>$f_3(1,\ldots,1) = 0$.</td>
</tr>
<tr>
<td>Sphere</td>
<td>$f_4(x) = \sum_{i=1}^{n} x_i^2$</td>
<td>[-5.12, 5.12]</td>
<td>$f_4(0,\ldots,0) = 0$.</td>
</tr>
</tbody>
</table>

Appendix B

1. **g01:**
   Minimize $f(x) = 5 \sum_{i=1}^{4} x_i - 5 \sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$

Subject to:

\[
g_1(x) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0
\]
\[
g_2(x) = 2x_1 + x_3 + x_{10} + x_{12} - 10 \leq 0
\]
\[
g_3(x) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0
\]
\[
g_4(x) = -8x_1 + x_{10} \leq 0
\]
\[
g_5(x) = -8x_2 + x_{11} \leq 0
\]
\[
g_6(x) = -8x_3 + x_{12} \leq 0
\]
\[
g_7(x) = -2x_4 - x_5 + x_{10} \leq 0
\]
\[
g_8(x) = -2x_6 - x_7 + x_{11} \leq 0
\]
\[
g_9(x) = -2x_8 - x_9 + x_{12} \leq 0
\]
\[
0 \leq x_i \leq 1 \quad (i=1,2,\ldots,9) , \quad 0 \leq x_i \leq 100 \quad (i=10,11,12) , \quad 0 \leq x_{13} \leq 1
\]

The optimum value is $f(x^*) = -15$ at $x^* = (1,1,1,1,1,1,1,1,3,3,3,3,3,3,3)$

Constraints $g_1, g_2, g_3, g_4, g_5, g_6, g_7, g_8, g_9$ are active.

2. **g02**

Maximize $f(x) = \frac{\sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i)}{\sum_{i=1}^{n} i^2 x_i^2}$

Subject to:

\[
g_1(x) = 0.75 - \prod_{i=1}^{n} x_i \leq 0
\]
\[
g_2(x) = \sum_{i=1}^{n} x_i - 7.5n \leq 0
\]
\[
0 \leq x_i \leq 10 \quad (i=1,2,\ldots,n) , \quad n = 20
\]

The optimum value is unknown. The known best value is $f(x^*) = 0.803619$

Constraint $g_1$ is active.

3. **g03**

Minimize $f(x) = -\sqrt{n} \prod_{i=1}^{n} x_i$

Subject to:

\[
h_1(x) = \sum_{i=1}^{n} x_i^2 - 1 = 0
\]
\[
0 \leq x_i \leq 10 \quad (i=1,2,\ldots,n)
\]
The optimum value is \( f(x^*) = -1 \) at \( x^* = (1/\sqrt{n}) \), \( n = 10 \).

4. **g04**

Minimize \( f(x) = 5.3578547x_1^2 + 0.8356891x_1x_2 + 37.293239x_1 - 40792.141 \)

Subject to:
\[
\begin{align*}
g_1(x) &= 85.334407 + 0.0056858x_2x_3 + 0.0006262x_1x_4 - 0.0022053x_3x_5 \\
g_2(x) &= 80.51249 + 0.0071317x_2x_3 + 0.0029955x_1x_2 + 0.0021813x_3^2 \\
g_3(x) &= 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 \\
0 &\leq g_1(x) \leq 92 \\
90 &\leq g_2(x) \leq 110 \\
20 &\leq g_3(x) \leq 25 \\
78 &\leq x_1 \leq 102, \ 33 \leq x_2 \leq 45, \ 27 \leq x_i \leq 45 \ (i = 3, 4, 5).
\end{align*}
\]

The optimum value is \( f(x^*) = -30665.539 \) at \( x^* = (78.33, 29.99525602568245, 36.775812905788) \).

5. **g05**

Minimize \( f(x) = 3x_1 + 0.000001x_1^3 + 2x_2 + (0.000002 / 3)x_2^3 \)

subject to:
\[
\begin{align*}
g_1(x) &= -x_4 + x_3 - 0.55 \leq 0 \\
g_2(x) &= -x_3 + x_4 - 0.55 \leq 0 \\
h_3(x) &= 1000\sin(-x_3 - 0.25) + 1000\sin(-x_4 - 0.25) + 894.8 - x_1 = 0 \\
h_4(x) &= 1000\sin(x_3 - 0.25) + 1000\sin(x_4 - 0.25) + 894.8 - x_2 = 0 \\
h_5(x) &= 1000\sin(x_4 - 0.25) + 1000\sin(x_3 - x_4 - 0.25) + 1294.8 = 0 \\
0 &\leq x_i \leq 1200 \ (i = 1, 2), \ -0.55 \leq x_2 \leq 0.55 \ (i = 3, 4).
\end{align*}
\]

The optimum value is \( f(x^*) = 5126.4981 \) at \( x^* = (679.9463, 1026.067, 0.1188764, -0.3962336) \).

6. **g06**

Minimize \( f(x) = (x_1 - 10)^3 + (x_2 - 20)^3 \)

Subject to:
\[
\begin{align*}
g_1(x) &= -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \leq 0 \\
g_2(x) &= (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \leq 0 \\
13 &\leq x_1 \leq 100, \ 0 \leq x_2 \leq 100
\end{align*}
\]

The optimum value is \( f(x^*) = -6961.81388 \) at \( x^* = (14.095, 0.84296) \).

**Appendix A**

(a) \( F_1 \): Optimal Capacity of Gas Production Facilities [20]

This is the problem of determining the optimum capacity of production facilities that combine to make an oxygen producing and storing system. Oxygen for basic oxygen furnace is produced at a steady state level. The demand for oxygen is cyclic with a period of one hour, which is too short to allow an adjustment of level of production to the demand. Hence the manager of the plant has two alternatives.

He can keep the production at the maximum demand level; excess production is lost in the atmosphere.

He can keep the production at lower level; excess production is compressed and stored for use during the high demand period.

The mathematical model of this problem is given by:

Minimize: \( f(x) = 61.8 + 5.72x_1 + 0.2623[(40 - x_i) \ln(x_i/200)]^{0.85} + 0.087(40 - x_i) \ln(x_i/200) + 700.23x_i^{-0.75} \)

Subject to: \( x_i \geq 17.5, x_i \geq 200; 17.5 \leq x_i \leq 40, 300 \leq x_i \leq 600 \)
(b) \( F_2 \): Optimal Thermo hydraulic Performance of an Artificially Roughened Air Heater [21]

In this problem the optimal thermo hydraulic performance of an artificially roughened solar air heater is considered. Optimization of the roughness and flow parameters (\( p/e, e/D, \text{Re} \)) is considered to maximize the heat transfer while keeping the friction losses to be minimum. The mathematical model of this problem is given by:

Maximize \( L = 2.51 * \ln e^+ + 5.5 - 0.1R_M - G_H \)

where \( R_M = 0.95x_2^{0.53}, G_H = 4.5(e^+)^{0.28} \)

\( e^+ = x_3T \left( f_2 + f_r \right)/2, f_2 = 0.079x_3^{-0.25}, f_r = 2(0.95x_3^{0.53} + 2.5* \ln(1/2x_1)^2 - 3.75)^{-2} \)

Subject to: \( 0.02 \leq x_i \leq 0.8, 10 \leq x_i \leq 40, 3000 \leq x_i \leq 20000 \)

(c) \( F_3 \): Design of Gear Train [22]

This problem is to optimize the gear ratio for the compound gear train. This problem shown in Figure 5 was introduced by Sandgren [32]. It is to be designed such that the gear ratio is as close as possible to 1/6.931. For each gear the number of teeth must be between 12 and 60. Since the number of teeth is to be an integer, the variables must be integers. The mathematical model of gear train design is given by,

\[
\text{Minimize } f = \left( \frac{1}{6.931} - \frac{T_aT_b}{T_aT_f} \right)^2 = \left( \frac{1}{6.931} - \frac{x_1x_2}{x_3x_4} \right)^2
\]

Subject to: \( 12 \leq x_i \leq 60 \quad i = 1,2,3,4 \)

\([x_1, x_2, x_3, x_4] = [T_a, T_b, T_a, T_f] \), \( x_i \)'s should be integers. \( T_a, T_b, T_d, \text{ and } T_f \) are the number of teeth on gears A, B, D and F respectively. The design of a compound gear train is shown in Figure 2.

![Compound Gear Train](Image)

(d) \( F_4 \): The lennard-jones atomic cluster problem [23]

The Lennard-Jones (LJ) model of inert gas cluster has been investigated intensively as a challenging testing ground for global optimization algorithms. The LJ problem can be formulated as follows: Let N atoms be given in three dimensional space. Let \( x^i = (x_1^i, x_2^i, x_3^i) \) represent the coordinates of atom \( i \). Let \( X = (x^1)^T, ..., (x^N)^T \) the LJ potential energy function \( v(r_{ij}) \) of a pair of atoms \( (i, j) \) is given by

\[
v(r_{ij}) = \frac{1}{r_{ij}^6} - \frac{1}{r_{ij}^2}, 1 \leq i, j \leq N \text{ where } r_{ij} = \|x^i - x^j\|
\]

The LJ potential function for a single pair of neutral atoms is a simple unimodal function. This is illustrated by Figure 3. It is easy to find the overall minimum of this function that is assumed at 1 with energy \(-1\). In a complex system, many atoms interact and we need to sum up the LJ potential functions for each pair of atoms in a cluster. The result is a complex energy landscape with many local minima. It is given by:

\[
\text{Minimize } V(X) = \sum_{i,j} v(\|x^i - x^j\|) = \sum_{i = 1}^{N} \sum_{j = 1}^{N} \left( \frac{1}{\|x^i - x^j\|^6} - \frac{1}{\|x^i - x^j\|^2} \right)
\]

As it is known the LJ problem is a highly nonlinear, non convex function with numerous local minimum that takes the LJ problem a challenging standard test problem for the global optimization algorithms. For illustration, we fix one atom’s position and let the others move around the fixed one. A cluster of more than 20 atoms has many of local minima along its LJ PES.
(e) $F_5$: Gas transmission compressor design [20]

Minimize $f(x) = 8.61 \times 10^5 \times x_1^{1/2} x_2 x_3^{-2/3} (x_2^2 - 1)^{-1/2} + 3.69 \times 10^4 \times x_3 + 7.72 \times 10^4 \times x_1^{-1} x_2^{0.210} - 765.43 \times 10^5 \times x_1^{-1}$

Bounds: $10 \leq x_1 \leq 55$, $1.1 \leq x_2 \leq 2.10$, $x_3 \leq 40$

Tarun Kumar Sharma: Assistant Professor in Amity Institute of Information Technology, Amity University Rajasthan, India. He has an experience of 11 years (4 years of research and 7 years of Academics). His research areas are evolutionary algorithms and their applications in Software Engineering. He is in Editorial Board and reviewer of many refereed Journals. He has published about 40 research papers in Journal of repute and in refereed international Journals.

Millie Pant: Associate Professor, Indian Institute of Technology, Roorkee, India. She has published above 200 research publications in referred journals and international conferences. She has been keynote speakers to various seminars, conferences and development programs. Her key research areas are Evolutionary Computing, Swarm Intelligence and their application in various areas of Engineering.