Developing an Intelligent Interactive Approach for Multi-objective Optimization Problems

Alia Youssef Gebreel

Abstract— This paper presents three intelligent algorithms, and combines all of them for solving some interactive multi-objective optimization problems. The hybrid optimization model combines Genetic Algorithm and Bacterial Foraging Optimization (BFO) with Harmony Search (HS) algorithm. This model is being designed and implemented in the hope to improve performance of GA and help BFO to escape from the local minima or maxima. Where, genetic algorithm is search based on the concepts of natural selection and genetics. BFO simulated the life cycles and the foraging behaviors of bacterial, which are called Escherichia coli. HS originally was inspired by the analogy between music improvisation and the optimization process. Also, a comparative study is presented to clarify the development approach based on the resulted solution's distance from the utopia point.

Index Terms— Genetic algorithm, Bacterial foraging optimization algorithm, Harmony search algorithm.

1 Introduction

THE classical interactive multi-criterion optimization methods demand the decision-makers to suggest a reference direction or reference points or other clues [5] which result in a preferred set of solutions on the Pareto-optimal front. It is often necessary for finding a single solution (or solutions) to optimization problem with conflicting criteria.

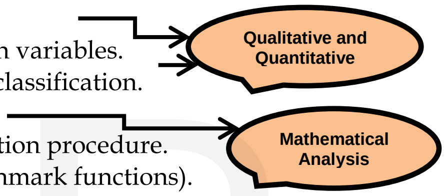
In the 20th century, artificial intelligence (AI) was one of the cutting-edge research fields. Over the second half period of this century, many methodologies have been investigated to explore the similarity between natural evolution and problem-solving algorithms, such as genetic algorithm, evolution strategies, evolutionary programming, genetic programming, particle swarm, and probabilistic model building genetic algorithm or estimation of distribution algorithm are based on the principle of evolution (survival-of-the-fittest) and imitate some natural phenomena (genetic inheritance) [3, 6, 12].

Artificial intelligence tools may be classified into the following models:

- Genetic Algorithms.
- Expert System.
- Decision Support System.
- Fuzzy Logic Expert system.
- Artificial Neural Networks.
- Neuro-Fuzzy Systems.
- Simulated Annealing.
- Ant-Colony Optimization.
- Swarm Optimization.
- DNA Computing.
- Artificial Immune System.
- Bacteria Foraging Optimization.
- Harmony Search Algorithm.

The corner stones of the optimization Model are as follows:

1- Problem definition.
2- Defining the decision variables.
3- Data collection and classification.
4- Model developing.
5- Developing the solution procedure.
6- Model testing (Benchmark functions).
7- Model implementation.
8- Feedback to improve the model performance.



The relationship between artificial intelligence and optimition is represented in the following figure [11].
To further enhance the optimization performances, an adaptive hybrid artificial intelligent algorithm will be studied in this research.

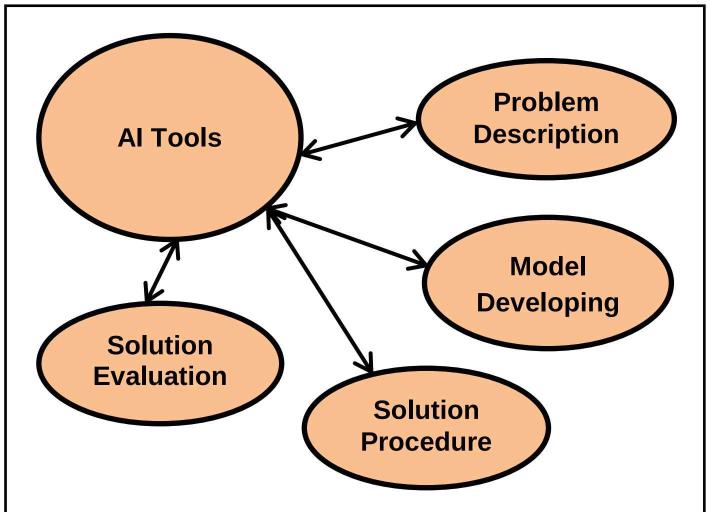


Fig. (1): Relationship between AI and optimization.

Alia [1] presented an overview of GA, BFA, and HA. The purpose of this paper is to introduce a novel approach, which integrates the genetic algorithm, and bacterial foraging algorithm with the harmony algorithm to find a preferred solution.

- *Alia Youssef Gebreel Mohamed is currently having Ph. D. degree in Operations Research, from Institute of Statistical Studies and Research (ISSR) at Cairo University, Cairo, Egypt, 2018.*

It is accomplished by testing these algorithms on some different examples. The integrated approach outperforms than applying every one of these three algorithms. Moreover, it has better performance for finding a best solution compared to the others reported in literature of the multi-objective optimization problems. Where, it provides simple ways to generate efficient solutions for the decision maker in the multi- objective problems. Thus, the decision maker can select his or her favorable efficient solution. Here, the decision maker interacts with computer program.

The major procedures of any one of these models can be expressed as follows:

1. Generate an initial population (t) based on the decision maker.
2. Calculate the fitness of population (t).
3. Repeat
   i. Select a new solution(s) (population(t+1)) from population(t) with preferring the fitter ones
   ii. Recombine solution(s) of population (t) to create a new population (t+1)
   iii. Perform Mutation of population (t+1)
   iv. Determine the fitness of population (t+1)
4. Stop until the best solution is good enough for the decision maker.

The rest of the paper is organized as follows. Section 2 presents the proposed interactive algorithm in detail. To show the effectiveness of this algorithm, four different numerical examples are illustrated in Section 3, followed by experimental results and analyses are shown in Section 4. Conclusion is also provided towards the end in Section 5.

2 THE PROPOSED ALGORITHM

The classical genetic and bacterial foraging optimization algorithms may take a time to reach a favorable efficient solution exactly for the decision maker. Where, the GA has no guarantee to find the favorable solution. But, it is possible to get an approximately favorable solution because GA is stochastic algorithm. Also, BFO algorithm may get struck in local optima. To resolve this problem, the harmony algorithm is integrated to them. This modified version is more efficient and accuracy for optimization process than applying every one of these three algorithms. Also to achieve this purpose, a programming in MATLAB code [8] has been developed in order to obtain good convergence and more accurate results of applications with save time. This algorithm is design to generate preferred solution. The decision maker determines a favorable efficient solution based on the distance of objective functions from the utopia point.

2.1 The Steps of the Proposed Interactive Algorithm

The basic steps of the proposed interactive algorithm can be summarized as follows:

1- Starting with an initial solution based on preference of the decision maker; this initial solution can be selected from any one of the individual optimal solutions in GA or all them for BFO and HS. Selecting the values of algorithm parameters. These values are taken after carrying out several trials on the example.

2- Using penalty functions to handle constrained problems in order to force the search towards feasibility in the genetic and harmony models.

3- Constructing an interactive computer programming GA.

4- Running program to obtain a new solution.

5- Evaluating the obtained solution. If the new solution is acceptable, stop.

6- Otherwise, constructing an interactive computer programming BFO with the output of GA as initial solution.

7- Update this initial solution with a suitable way for BFO to give a new solution.

8- If the new solution is acceptable, stop. Otherwise, combining HA procedure with BA to extract the preferred solution directly based on the final solution of BA (after a suitable updating for HA).

9- When initialize the optimization problem and algorithm parameters to minimize the objective functions, the specification of each decision variable is design as possible value range for each decision variable. The values of algorithm parameters are selected on the basis of empirical suggestions.

10- Calculating the objective functions value for the newly selected vector. If this value is better than the worst harmony vector in the harmony matrix, it is then included in the matrix, while the worst one is taken out of the matrix. The harmony memory matrix is then sorted in descending order by the objective function value.
Note that: In these models, no weighting coefficient is required.

11- Repeating Steps 9 and 10 until the pre-selected maximum number of iterations is reached. This number is selected for large enough cycles to observe that there is not any further improvement in the resulted solution.

12- If a preferred solution is obtained, stop. Otherwise, updating the HS algorithm by a new initial vector, which is selected from two resulted solutions, and continuing with step 9. This work focuses on the harmony search parameters with respect to the initial solution-setting-free technique.

By this way, hybridized genetic with bacterial foraging optimization and harmony search reduce the convergence time and enhance the accuracy.

2.2 The Proposed Algorithm's Pseudo-code

The pseudo-code as well as the flowchart (**Fig. (2)**) of the complete algorithm is presented below:

(1) Genetic Algorithm

Step (1-1): Initialize population and parameters of GA:

Define fitness function F(x) = ($f_1(\mathbf{x})$ + $f_2(\mathbf{x})$ + ... + $f_k(\mathbf{x})$) + penalty parameter × (sum of all the problem constraints), $k \geq 2$ of objective functions, the decision varibles $(\mathbf{x}) = (x_1, x_2, \ldots, x_n)T$, and n = the number of the decision variables.

Generate a new population of solutions (P):

Start with initial solutions $(\mathbf{x}) = (x_1, x_2, \ldots, x_n)^T$ based on decision maker's opinion (for example, it can determine by initial range as [minimum value of the individual optimal vectors : maximum value of the individual optimal vectors]).

Define parameters of GA:

There are several parameters which characterize GA:

Population size -- specifies how many individuals that are in each generation,

Population type -- specifies the data type of the input (double-vector, bit-string, and custom) to the fitness function,

Creation function (CreationFcn) -- specifies the function that creates the initial population to find the minimum of a function using the GA such as uniform, Gaussian, or custom,

Population initial range -- specifies the range of the vectors in the initial population that is generated by the creation function. The first row contains lower bounds for the entries of the vectors in the initial population, while the second row contains upper bounds,

Crossover function (CrossoverFcn), ***Crossover probability (pc)***, ***Crossover fraction*** (CrossoverFraction) in reproduction options -- specifies the fraction of the next generation,

Mutation function (MutationFcn), ***Mutation probability (pm)***, ***and Stopping criteria options that contain the following:***

Generations number (Generations) -- Specifies the maximum number of iterations the genetic algorithm will perform.

Time limit (TimeLimit) -- Specifies the maximum time in seconds the genetic algorithm runs before stopping.

Fitness limit (FitnessLimit) -- The algorithm stops if the best fitness value is less than or equal to the value of fitness limit.

Stall generations (StallGenLimit) -- The algorithm stops if there is no improvement in the best fitness value for the number of generations specified by Stall generations.

Stall time (StallTimeLimit) -- The algorithm stops if there is no improvement in the best fitness value for an interval of time in seconds that specified by Stall time.

Step 1-2. Find fitness of population: Evaluate the fitness function of each point for the population using the objective functions and constraints of the problem. Also, apply the penalty function method for constrained problems.

<u>Repeat on this generation until a favorable efficient solution is met</u>:

Step 1-3. Parent selection: select the better solutions from the old population of solutions by selection function.

Step 1-4. Perform the following genetic operators on selected parent:

1-4-1. Crossover operator: Apply crossover with probability (p_c) to parent for getting new solution.

1-4-2. Mutation operator: Apply mutation with probability (p_m) of existing solution to create new solution.

Step 1-5. Decode and fitness calculation:

Select ga function- Genetic Algorithm in the solver field of MATLAB software. The fitness function is called from the GA to determine the fitness of each solution string generated during the search.

Step 1-6. Survivor selection:

The survivor selection policy determines which individuals are to be kicked out and which are to be kept in the next generation.

Step 1-7. Termination test:

If the preferred solution is not obtained, return to **step 1-2**. Otherwise, stop and save this solution.

This process in an iterative manner is called generation until the termination criteria (which setting by decision maker) is reached.

(2) Bacterial Foraging Optimization Algorithm

Step 2-1 (a). Transport the local solution obtained from the GA to the BFO as an initial solution with a suitable value range in each decision variable by decision maker (**DM**). Or set new initial solution (with specification of each decision variable, and a possible value range in each decision variable) if GA closed to the preferred solution for the decision maker.

Step 2-1 (b). Initialize parameters:

Define fitness function $\mathbf{F(x)} = (f_1(\mathbf{x}) + f_2(\mathbf{x}) + \dots \dots + f_k(\mathbf{x}))$, $k \geq 2$ of objective functions, the decision varibles $(\mathbf{x}) = (x_1, x_2, \dots, x_n)^T$, and n = the number of the decision variables. The initial range for every variable is determined by a range for each bacterium based on the number of bacterial in the population. For example, if the number of bacterial = 4 and the number of variables = 2, then the initial solutions set as:

$x_1 = P_1 = (\text{value}_{11}; \text{value}_{12}; \text{value}_{13}; \text{value}_{14})'$;

$x_2 = P_2 = (\text{value}_{21}; \text{value}_{22}; \text{value}_{23}; \text{value}_{24})'$;

Note that the penalty function method to deal with a constrained problem is not considered here.

Define p, S, N_c , N_s , N_{re} , N_{ed} , P_{ed} ,C (i) ($i = 1, 2, \dots, S$), θ_i,

Where:

p: Dimension of the search space,

S: The number of bacteria. It is a positive even integer,

N_c: Number of chemotactic steps,

N_s: Number of swim steps (or maximum number of steps),

N_{re}: The number of bacteria reproductions (splits) per generation,

N_{ed}: The number of elimination and dispersal steps,

P_{ed}: The probability that each bacterium will be eliminated/ dispersed,

$C(i)$: The run-length unit (i.e., the chemotactic step size during each run or tumble). Here, the same run-length unit is used for all bacteria in the colony.

$J(i, j, k)$ = Fitness value or cost of *i-th* bacteria in the *j-th* chemotaxis and *k-th* reproduction steps.

$\theta(i, j, k)$= Position vector of *i-th* bacterium in *j-th* chemotactic step and *k-th* reproduction steps.

Jbest (j, k) = Fitness of best position in the *j-th* chemotaxis and *k-th* reproduction steps.

Jglobal= Fitness value or cost of the global best position in the entire search space.

Step 2-2. Elimination-dispersal loop: $l = l + 1$.

Step 2-3. Reproduction loop: $k = k + 1$.

Step 2-4. Chemotaxis loop: $j = j + 1$.

2-4-1. For $i = 1, 2\dots, S$, take a chemotactic step for bacterium i as follows.

2-4-2. Compute fitness function, $J(i, j, k, l)$.

2-4-3. Let J last = $J(i, j, k, l)$ to save this value since we may find better value via a run.

2-4-4. Tumble: generate a random vector $\Delta(i) \in R_p$ with each element $\Delta_m(i)$, $m = 1, 2, \ldots, p$, a random number on $[-1, 1]$.

2-4-5. Move: Compute $\theta_i(j+1, k, l)$.This results in a step of size $C(i)$ in the direction of the tumble for bacteria i.

2-4-6. Compute the fitness function $J(i, j+1, k, l)$ with $\theta_i(j+1, k, l)$.

2-4-7. Swim:

(i) Let m =0 (counter for swim length)

(ii) While $m < N_s$ (if not climbed down too long) do

a) Let $m = m+1$

b) If $J(i, j+1, k, l) < J_{\text{last}}$, define $J_{\text{last}} = J(i, j+1, k, l)$, Then, another step of size $C(i)$ in the same direction will be taken as:
$\theta_i(j+1, k, l) = \theta_i(j, k, l) + C(i) \dfrac{\Delta(i)}{\sqrt{\Delta^T(i)\,\Delta(i)}}$,and use the new generated $\theta_i(j+1, k, l)$ to compute the new $J(i, j+1, k, l)$.

c) Else Let $m = N_s$.

2-4-8. Go to next bacterium (i+1): if $i \neq S$ go to 2-4-2 to process the next bacterium.

Step 2-5. If $j < N_c$, go to Step 2-4. In this case, continue chemo-taxis since the life of the bacteria is not over.

Step 2-6. Reproduction:

For the given k and l, and for each i =1, 2, ... , S, let
$J_{i\ \text{health}} = \sum_{j=1}^{Nc+1}(i, j, k, l)$ be the health of the bacteria. Sort bacterium in order of ascending values (J_{health}).

The S_r bacteria with the highest J_{health} values die and the other S_r bacteria with the best values split, and the copies that are made are placed at the same location as their parent.

Step 2-7. If $k < N_{re}$ go to Step 2-2. In this case, the number of specified reproduction steps is not reached; start the next generation in the chemotactic loop.

Step 2-8. Elimination-dispersal: for i =1, 2, ... ,S, with probability P_{ed}, eliminate and disperse each bacterium, which results in keeping the number of bacteria in the population constant. To do this, if a bacterium is eliminated, simply disperse one to a random location on the optimization domain. If $l < N_{ed}$, then go to Step 2-6, otherwise, end.

(3) Harmony Search Algorithm

Step 3-1 (a). Transport the local solution obtained from the GA or BFO to the HS as **an initial solution with a suitable value range in each decision variable** by decision maker (**DM**). Or set new initial solution (with specification of each decision variable, and a possible value range in each decision variable) if GA or BFO closed to the preferred solution for the decision maker. For example, if the number of the harmony memory size = 4 and the number of variables = 2, then the initial solutions set as:

[The values of the first variable; the values of the second variable] = [(value$_{11}$ value$_{12}$ value$_{13}$ value$_{14}$); (value$_{21}$ value$_{22}$ value$_{23}$ value$_{24}$)].

Step 3-1 (b). Set the parameters and initialize the HM:
Define fitness function $\mathbf{F(x)} = (\ f_1(\mathbf{x}) + f_2(\mathbf{x}) + \ldots \ldots + f_k(\mathbf{x})\ )$ + penalty parameter × (sum of all the problem constraints), $k \geq 2$ of objective functions, the decision varibles $(\mathbf{x}) = (x_1, x_2, \ldots, x_n)^T$, and n = the number of the decision variables.

Where, the brief sub-pseudo-code of fitness function based on penalty function method for constrained problem has been provided below:

```
Function sum =Fitness (x)
Sum = cg (x)
Sum = ( $f_1(x) + f_2(x) + \ldots \ldots + f_k(x)$ ) + penalty (cg (x))
end
function constraints g(x) > 0
Constraint (1), Constraint (2), Constraint (3), …. , Constraint (m)
Sum = 0;
for i=1: number of inequality constraints= m
if (gx (i) < 0)
Sum = Sum ± penalty's parameter × gx (i);
end
end
End
```

Define (HMS), (HMCR), (PAR min), (PAR max), (bw min), (bw max).

Define the maximum number of iterations (NI).

Step 3-2. Generate initial population (real number array (HM)):

min = minimum visible value.

max = maximum visible value.

Step 3-3. Improvise a new harmony $\mathbf{x}_{new}$ as follows:

```
while (Stop condition (current iteration ≤ NI)) do
  for all decision variables (i = 1, 2, …, n) do
   If (rand ∈ (0, 1) < HMCR) then choose a value of one of the solutions from the harmony memory (HM) for i
     If (rand ∈ (0, 1) < PAR) then adjust the value of i by:
       $x_{new}(i) = x_{old}(i) \pm rand \in (0, 1) \times bw$
    end if
       else (with probability 1- HMCR) use a random value for this decision variable
end if
 end for
```

Step 3-4. Update the HM (minimization objectives)**:**

```
  If the new solution is better than the worst solution in the harmony memory then accept the new harmony and replace the worst solution with it.
  end if
end while.
```

Step 3-5. Find the current best solutions:

If termination criterion is reached, return the best solution in the harmony memory; otherwise update the initial population and go to **Step 3-3**.

End.

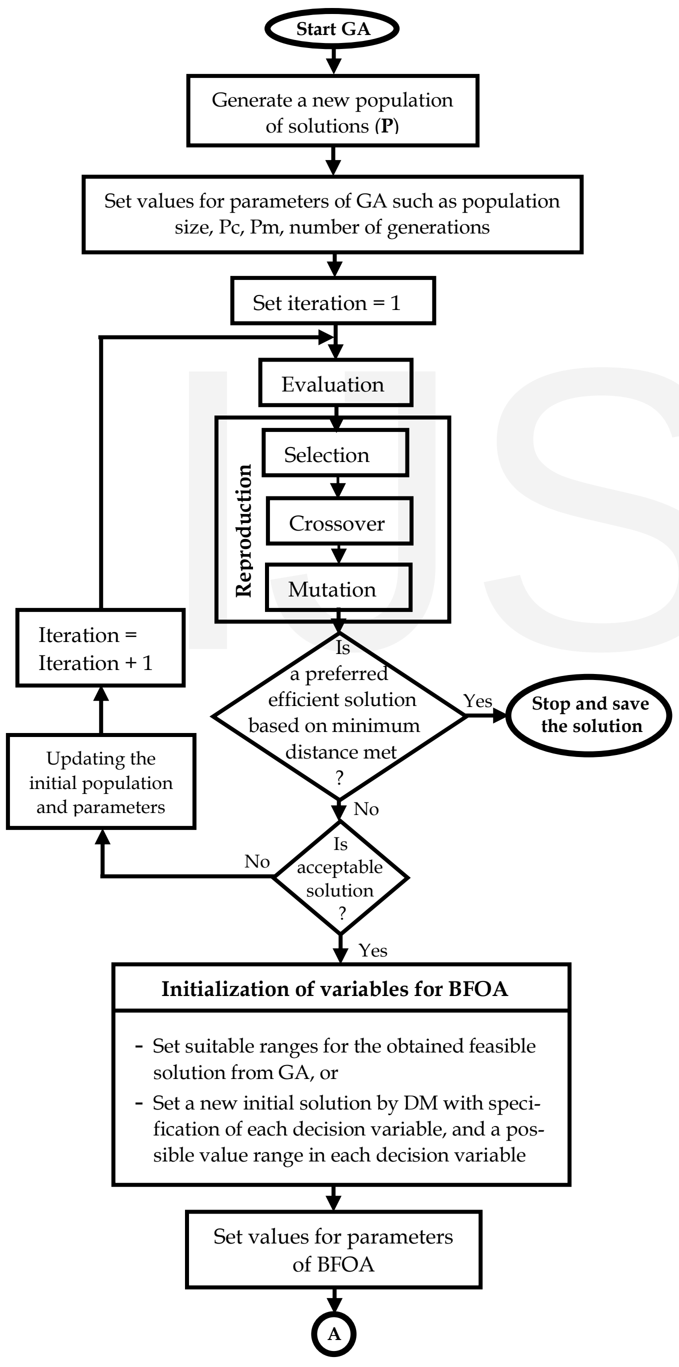
Note that:

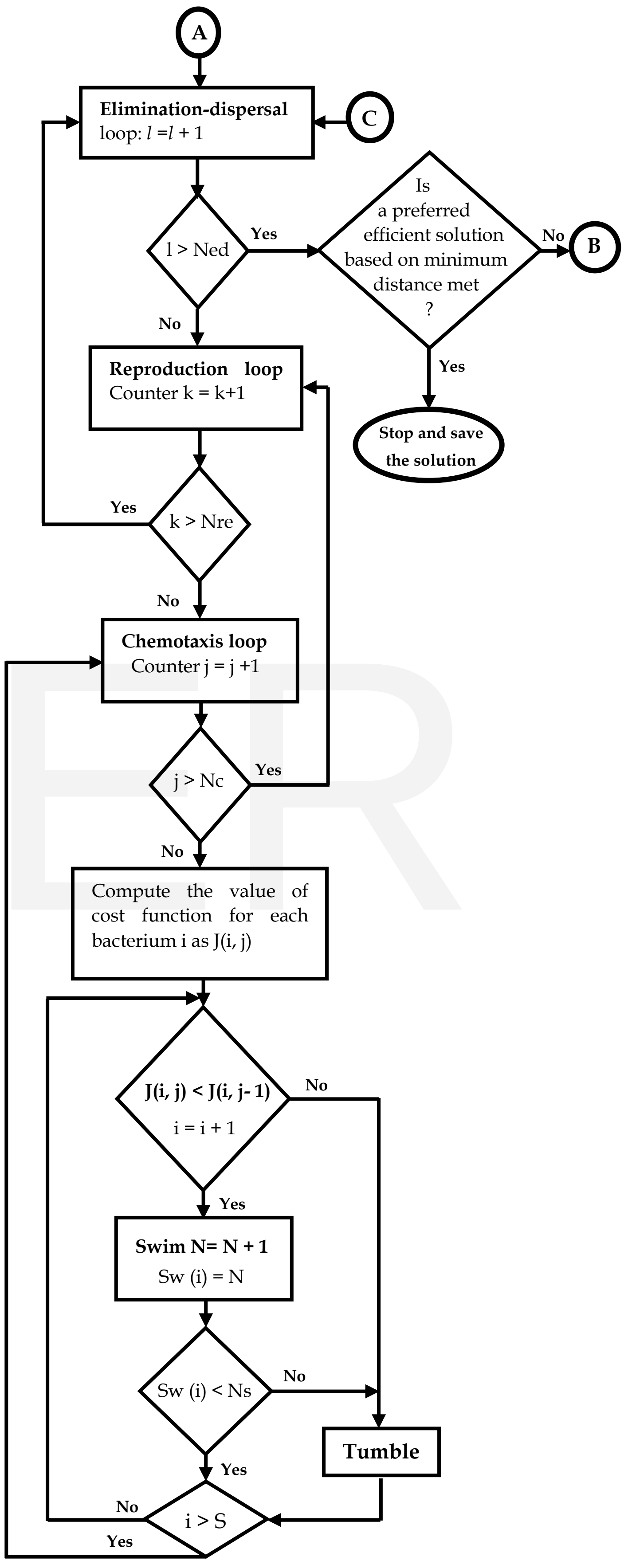
1- The acceptable solution may be infeasible but have less distance.

2- After each generation of GA, BFO, and HS, a typical and popular Euclidean distance is employed to measure the distance between the utopia vector and a resulted solution vector. Euclidean distance is defined as the straight-line distance between two points. For *N*-dimensional space, the Euclidean distance between these two points' *Ui* and *Ai* is given by:

$$ED=\sqrt{\sum_{i=1}^{N}(Ui-Ai)2} \quad \textbf{(1)}$$

Where, *Ui* (or *Ai*) is the coordinate of *U* (or *A*) in dimension *i* [9]. The selected solution has the minimum Euclidean distance.


Start GA
Generate a new population of solutions (P)
Set values for parameters of GA such as population size, Pc, Pm, number of generations
Set iteration = 1
Evaluation
Reproduction
Selection
Crossover
Mutation
Is a preferred efficient solution based on minimum distance met ?
Yes
Stop and save the solution
No
Is acceptable solution ?
No
Updating the initial population and parameters
Iteration = Iteration + 1
Yes
Initialization of variables for BFOA
- Set suitable ranges for the obtained feasible solution from GA, or
- Set a new initial solution by DM with specification of each decision variable, and a possible value range in each decision variable
Set values for parameters of BFOA
A


A
Elimination-dispersal loop: l = l + 1
C
l > Ned
Yes
Is a preferred efficient solution based on minimum distance met ?
No
B
Yes
Stop and save the solution
No
Reproduction loop Counter k = k+1
k > Nre
Yes
No
Chemotaxis loop Counter j = j +1
j > Nc
Yes
No
Compute the value of cost function for each bacterium i as J(i, j)
J(i, j) < J(i, j- 1)
i = i + 1
No
Yes
Swim N= N + 1
Sw (i) = N
Sw (i) < Ns
No
Yes
Tumble
i > S
No
Yes

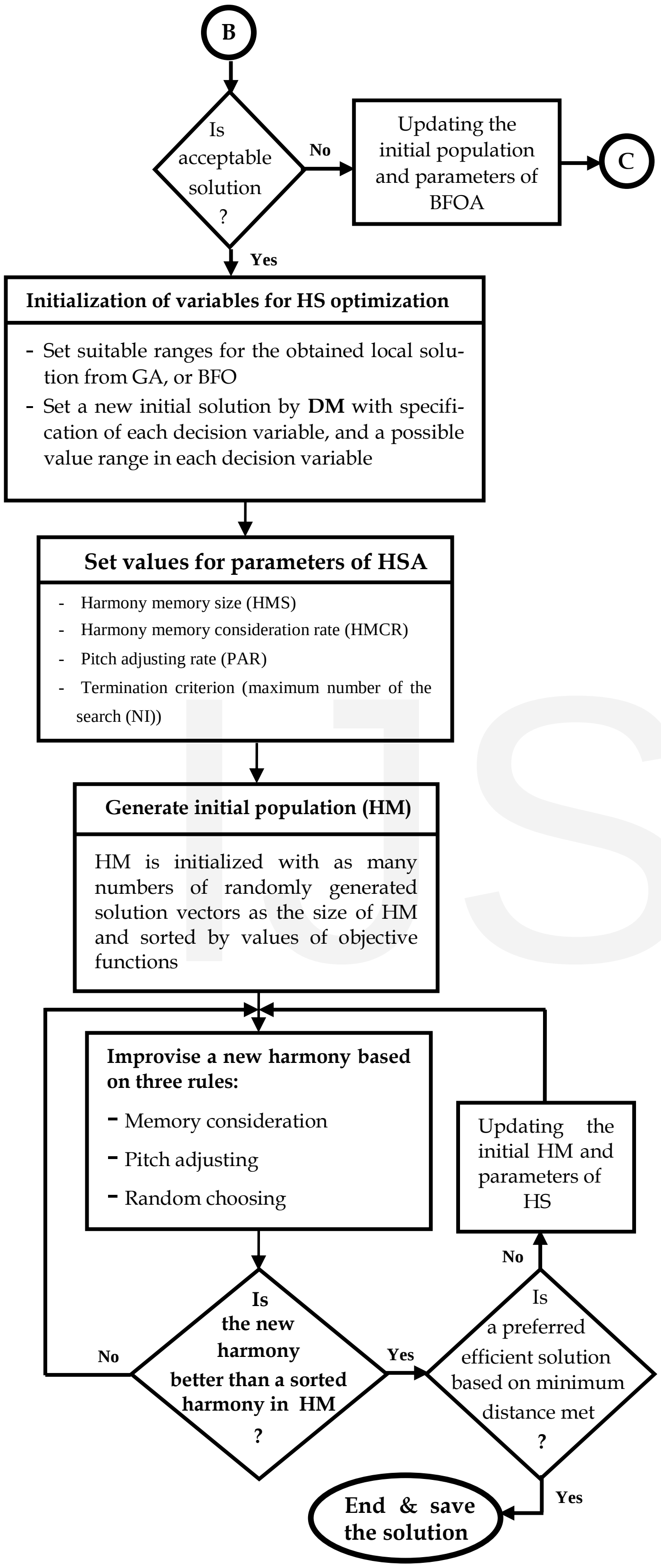

B
Is acceptable solution ?
No
Updating the initial population and parameters of BFOA
C
Yes
Initialization of variables for HS optimization
- Set suitable ranges for the obtained local solution from GA, or BFO
- Set a new initial solution by DM with specification of each decision variable, and a possible value range in each decision variable
Set values for parameters of HSA
- Harmony memory size (HMS)
- Harmony memory consideration rate (HMCR)
- Pitch adjusting rate (PAR)
- Termination criterion (maximum number of the search (NI))
Generate initial population (HM)
HM is initialized with as many numbers of randomly generated solution vectors as the size of HM and sorted by values of objective functions
Improvise a new harmony based on three rules:
- Memory consideration
- Pitch adjusting
- Random choosing
Updating the initial HM and parameters of HS
No
Is the new harmony better than a sorted harmony in HM ?
No
Yes
Is a preferred efficient solution based on minimum distance met ?
Yes
End & save the solution

Fig. (2): Flowchart of the proposed solution algorithm.

3 Comparative Study

In this section, some different multi-objectives examples are used to test the proposed algorithm's performance.

The code of these examples has been written with MATLAB 7.0. The key factor in these applications is how the range of initial point is selected according to the problem with controlling algorithm's parameters. Also here, the preferred solution is selected from these algorithms according to their non-dominance based on its distance from the utopia point.

Example (3.1):

In this example, we want to minimize two objectives, each having one decision variable [4].

Min: $(f_1 = (x+2)^2 - 10,\quad f_2 = (x-2)^2 + 20)$,

Subject to: $-1.5 \le x \le 0$.

The following graph plots two objective functions on the same axis.

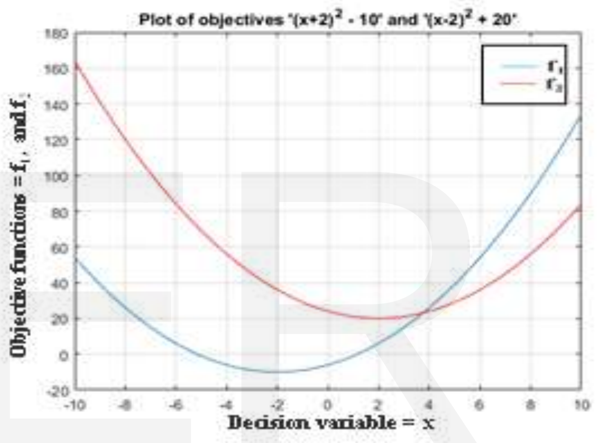


Fig. 3: Plot of objectives for example (3.1).

The two objectives have their minima at $x^*= -2$ and $x^*= +2$ respectively. However, in a multiobjective problem, x= -2, x= 2, and any solution in the range $-2 \le x \le 2$ is Pareto- optimal. It is noted in [4] that there is no single solution to this multiobjective problem. But, the goal of the proposed algorithm is to find preferred solution in that range (ideally with a good spread).

The Genetic Algorithm solver assumes the fitness function will take one input x. The following fitness function computes the value of each objective function and returns these values in a single vector output.

Fitness Function =

@ (x) ((x(1)+ 2)^2- 10+ (x(1)- 2)^2+20). **(2)**

The parameter settings for GA are as follows:

Population size: 20.0, Fitness limit: 10.0,

Crossover fraction: 0.7, Mutation function: uniform,

Stall generation limit and stall time limit are infinity,

The first population initial range is [-2: 2].

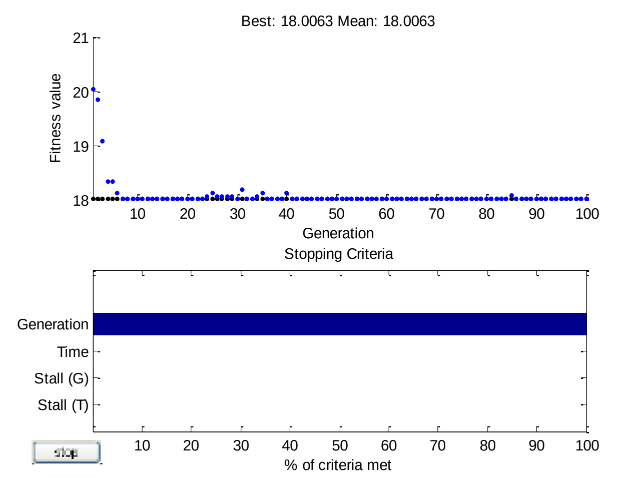


Fig. 4: The first generation of GA for example (3.1).

By direct search toolbox of MATLAB code 7.0, you can use GAOPTIMSET for default GA options structure. The above figure shows the converging process of the fitness values over 100 steps. The results of the first generation are:

x = -0.0561, the first objective is -6.221, the second objective is 24.2275, and their distance from the utopia point is 5.6702.

By followed population initial ranges ([-0.0037: 2.00], [-0.0027:0.0123], [-4.3982e-004: 2.4134e-004], [-9.8796e-007: 7.2198e-006], [-7.9727e-008: 9.0011e-009], [-2.1705e-010: 3.9561e-010], [-7.1521e-011: 8.7339e-012]), the solution of GA at last generation is as follows:

***x* = 2.6221e-012**, the first objective is -6.0, the second objective is 24.0, the best fitness (that means the optimal solution of the problem at equal weights) as seen in the following figure is 18.0 and their distance from the utopia point is 5.6569.

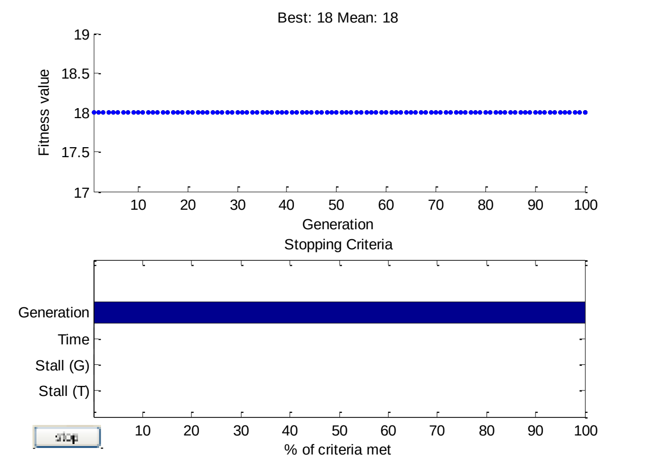


Fig. 5: The last generation of GA for example (3.1).

This solution of GA has the global minimum at the origin or very near to the origin. As can be seen from following figure, there is good convergence of the Pareto front. Preferred solution is obtained on the minimum convergence of the Pareto front.

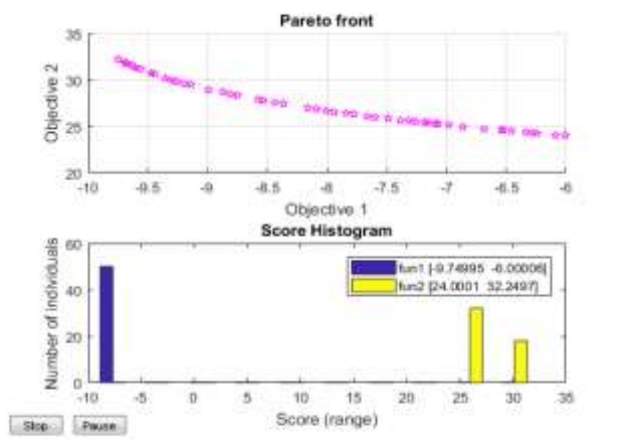


Fig. 6: Experimental results for example (3.1).

The second sub algorithm to minimize these objectives with satisfying their constraints is ***bacterial forging optimization***. It studies the behavior of bacteria in a given search space. BFA is coded as M-file in MATLAB platform.

Step 1-a: Bacteria Representation and initialize parameters:

Each bacterium's position represents one possible solution required for the problem. The number of dimensions of search space is p. In each dimension of search space, bacteria position is [0.0; 1.0]' for next generation. In the each iteration of chemotaxis step, each bacterium tumbles to the new random position. Position of ith bacterium in jth chemotaxis and kth reproduction step is defined as:

x= P(1,: ,j+1,K, *l*); **(3)**

For initialization, we must choose p, S , Nc , Ns , Nre , Ned , Ped , and the C(i), i=1,2…… S. Calculations are restricted within specified search ranges with the different parameters as given in table **1**. There two cases of the initial search ranges. Case (1) is based on the output of GA, but case (2) is based on free selected initial point. The other parameters used for BFO-based trial and error selection are shown in table **1**.

Step 1-b: Define and evaluate the fitness function of the algorithm:

In each generation, each bacterium is evaluated, and a value of *goodness* or *fitness* is returned by a fitness function. This evolution is driven by the fitness function.

The fitness function =

((x(S)+ 2)^2 -10 + (x(S)- 2)^2 + 20**)**. **(4)**

Step 2: Execute the bacterium chemotaxis cycle.

Step 3: Run the bacterial dispersal cycle.

Step 4: Evaluate and select the best solution:

After each generation of BFO, the Euclidean distance is employed to measure the distance between the utopia vector and the resulted solution vector.

Obviously when the bacterium has smaller C(i) = 0.00001, it is closed to its start point and not able to escape from it. But, the bacterium with larger C(i) = 0.1 can explore the whole search space and escape from these local solutions to enter the domain with the efficient solutions.

Some or all parameters may be change during the evolution to produce significant improvement in performance results of the algorithm.

With these values of controller BFO parameters that are chosen after a considerable number of trials and error, the output of BFO is $x = 0$, then the first objective is -6.0, the second objective is 24.0, and their distance from the utopia point is 5.6569. Then, the BFO gives better benefits when compared to GA.

Table 1. Control parameters of the BFA used in example (3.1)

Data	Range/ Value Case (1)	Range/ Value Case (2)
The initial solution	[-7.1521e-011; 0.0; 1.0; 2.0]	[0.0; 1.0]
S	4	2
Nc	1	1
Ns	1	1
Nre	1	1
Ned	2	2
Ped	0.1, 0.2	0.001, 0.1
The run-length C(i)	0.00001*ones(S,1)	0.0001*ones(S,1), 0.00001*ones(S,1)

In next phase, ***the harmony search algorithm*** is reinitialized at point (-7.1521e-011, 0.0, 1.0, 2.0) with 1000 run-length unit (NI) and HMS= 4. The sub-pseudo-code of the complete algorithm is presented below:

```
Start
Objective function, f(x) =
(f1 + f2) = ((x+ 2)^2 – 10 + (x- 2)^2 + 20)        (5)
Generate initial harmonics, [-7.1521e-011   0.0   1.0   2.0]
Define pitch adjusting rate, (PAR min= 0.04, PAR max= 0.09, bw min= 0.01, and bw max= 0.07)
Define harmony memory accepting rate (HMCR= 0.07)
While run-length < Max number of iterations (1000)
Generate new harmonics by accepting best harmonics
Adjust pitch to get new harmonics (solutions)
if (rand > HMCR= 0.07),
choose an existing harmonic randomly
else if (rand > pitch adjusting rate), adjust the pitch randomly within limits
else generate new harmonics via randomization
end if
Accept the new harmonics (solutions) if better
end while
Find the current best solutions
end
```

Finally for few times of run, the current best solution of HS is similar to BFO with less time search. This example has a unimodal variable, and only one minimum.

Example (3.2):

In this example, we consider two objectives, and two variables minimization optimization problem [2] as shown below to better illustrate the working of the GA, BFO, and HS.

Min: $(f_1 = x_1,\ \ f_2 = (1+x_2) / x_1)$,

Subject to: $0.1 \leq x_1 \leq 1$,

$0 \leq x_2 \leq 5$.

Where, $f_1^* = 0.1$ with ($x_1^* = 0.1$, $x_2^* = 0$), and $f_2^* = 1.0$ with ($x_1^* = 1.0$, $x_2^* = 0$).

Now we show the step by step procedure of the proposed algorithm.

A typical genetic algorithm procedure takes the following steps:

A population of candidate solutions is initialized as follows: Population initial range = **(**[0.1; 1.0], [0.1; **0.7912**], [0.1; **0.7754**], [0.1; **0.7663**], [0.1; **0.7602**], [0.1; **0.7430**], [0.1; **0.7403**], [0.1; **0.1026**], [**0.0**; **0.1000**], [0.0; **0.0015**]**)**; Population size = 20, 10, 6; Crossover fraction = 0.8, 0.5, 0.01, 0.001; Mutation function = Uniform; Fitness limit = 2; Stall generation limit = Stall time limit = infinity and Penalty parameter = 0.1. New solutions are created by applying genetic operators (mutation and/or crossover).

The fitness **(**@ (x) (x(1)+ ((x(2)+ 1)/ x(1)) + (Penalty parameter 0.10)*(x(1) + 0.1+ x(1) -1.0 + x(2) - 5))**)** of the resulting solutions are evaluated and suitable selection strategy is then applied to determine which solutions will be maintained into the next generation. The procedure is then iterated.

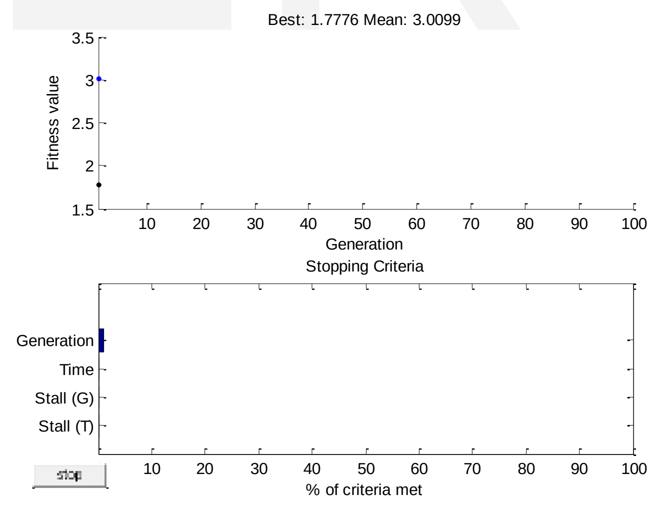


Fig. 7: The first generation of GA for example (3.2).

In this example, the penalty function is used to transform this constrained problem to unconstrained problem. The two following various graphs show GA process according to fitness evolution.

The first solution of GA is as follows:

$x_1 = 0.9551$, $x_2 = 0.1521$, $f_1 = 0.9551$, $f_2 = 1.2062$, total of two objectives = 2.1614, the distance from the utopia point = 0.8796. But, the last solution of GA is as follows: **x_1 = 0.7403, x_2 = 1.6554E-07, f_1 = 0.7403, f_2 = 1.3508**, total of two objectives= **2.0911**, the distance from the utopia point = **0.7301**.

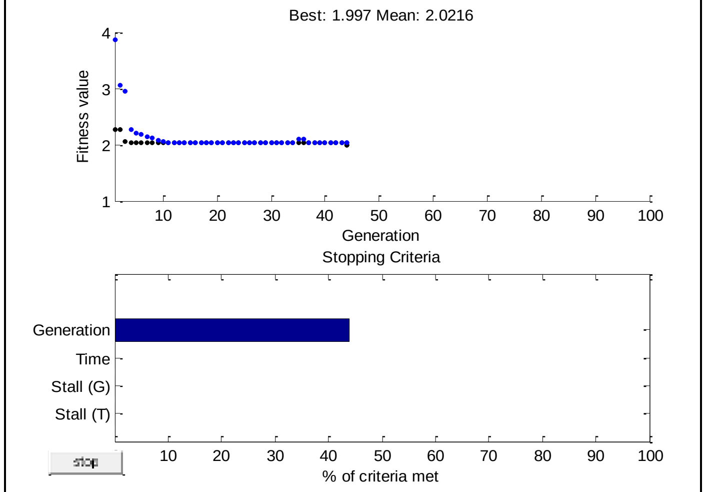


Fig. 8: The generation of GA when x_1= 0.7403 and x2= 0.2398 for example (3.2).

Following, the BFO process with the path of four bacteria that start at (0.1; 0.4; 0.7; 1.0), and (0.0; 0.00000016554; 0.1; 1.0) with different C(i). The parameter setting for BFO is S = 4, 2; Nc = 2, 3, 1; Ns = 1, 3, 2; Nre =1, 3; Ned = 1, 3, 2; Ped = 0.01, 0.1, 0.12; C(i) = 0. 001*ones (S,1), 0.01*ones (S,1) and 0.1*ones (S,1). **Fig. 8**, and **Fig. 9** illustrate the foraging process of the function values found by these bacteria.
The first solution of BFO is as follows:
x_1= 1.0004, x_2= 0.0004, f_1 = 1.0004, f_2 = 1.0, total OFs = 2.0004, C_1 = 1.0004, C_2 = 4.0987e-004, and distance = 3.9276e-004.
After some generations with the initial solutions (0.7409; **0.7374**) and (0.0000; 0.0000), the last solution of BFO is as follows: **x_1= 0.7403, x_2= 0.0000, f_1 = 0.7403, f_2 = 1.3508**, total OFs = **2.0911**, distance = **0.7301**.

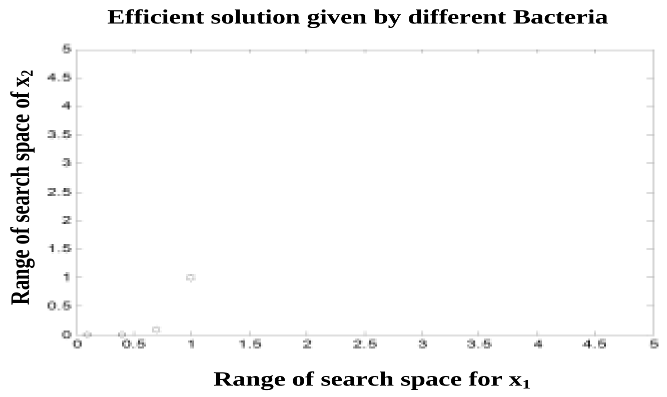


Fig. 9: The first solutions of BFO for example (3.2).

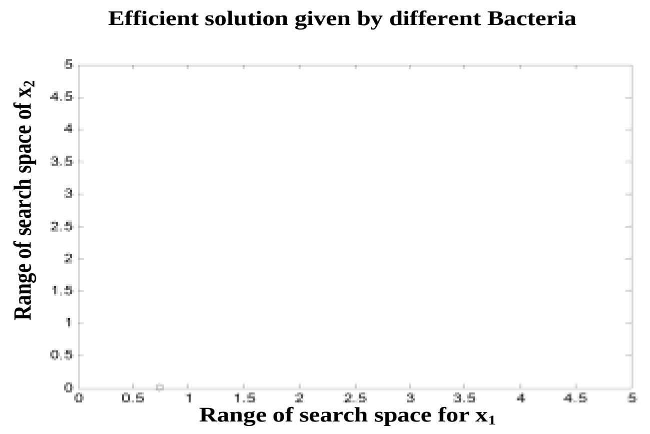


Fig. 10: The last solutions of BFO for example (3.2).

From the illustrated graphical and above results, the GA-BFO algorithm outperformed over genetic algorithm in terms of solution accuracy and convergence speed to get a favorable solution.
The following sub-algorithm **(HS)** starts with [0.1 0.4 0.7 1.0; 0.0 0.00000016554 0.1 1.0]; which is followed by [0.1 0.4 0.7 1.0; **0.0 0.0 0.0 0.0**], [**0.2140** 0.4 0.7 1.0; 0.0 0.0 0.0 0.0], [0.7 **0.7466**; 0.0 0.0], [**0.7397** 0.7420; 0.0 0.0], [0.7397 **0.7410**; 0.0 0.0], [**0.7401** 0.7407; 0.0 0.0], and [0.7401 **0.7404**; 0.0 0.0] in several iterations, respectively. The other parameters are setting as: NI**=** (7000, 1000), HMS= (4, 3, 2), HMCR= (0.07, 0.7, 0.02), PAR min= (0.04, 0.4, 0.07), PAR max = (0.09, 0.9), bw min = (0.01, 0.1, 0.05), bw max= (0.07, 0.7, 0.06), and Penalty parameter = (-0.0001). The resulted solution as graphing in the following figure is **x_1= 0.7403, x_2= 0.0000, f_1 = 0.7403, f_2 = 1.3508**, total OFs = **2.0911**, distance = **0.7301**.

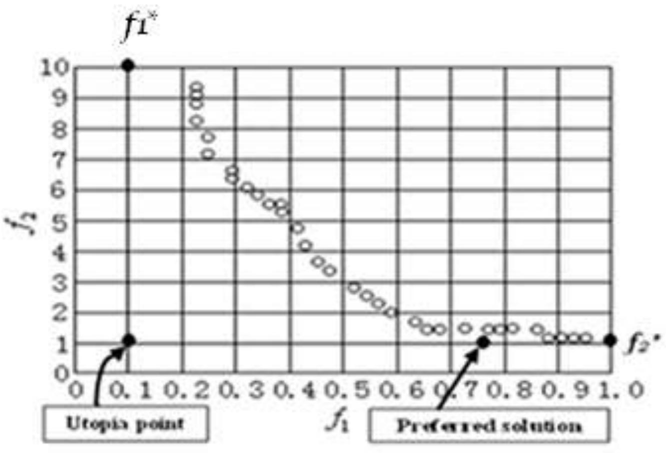


Fig. 11: Experimental results for example (3.2).

It is clearly seen from the above figure, even though there is good convergence and quite good diversity, there is the preferred solution that has minimum distance to the utopia point and it outsides the resulted Pareto front from [2]. The proposed BF–HS algorithm converges to the preferred solution much faster than the GA approach.

Example (3.3):

The example shows multi-objective convex non-linear programming problem. This problem was solved by an extended method of TOPSIS for the convex non-linear multi-objective problems in [7].

Max: $(f_1 = x_1^2 + x_2^2 + x_3^2,\ \ f_2 = (x_1 - 1)^2 + x_2^2 + (x_3 - 2)^2)$,

Min: $(f_3 = 2x_1 + x_2^2 + x_3)$

Subject to: $-x_1 + 3x_2 - 4x_3 + 6 \geq 0$,

$-2x_1^2 - 3x_2 - x_3 + 10 \geq 0$,

$x_1, x_2, x_3 \in R^3,\ 0 \leq x_1 \leq 3, 0 \leq x_2 \leq 4, 0 \leq x_3 \leq 2$.

Where, the individuals optimal are as follows:

$f_1^* = -11.1111$ with $(x_1^* = 0, x_2^* = 2.66667, x_3^* = 2.0)$, $f_2^* = -16.1111$ with $(x_1^* = 0, x_2^* = 3.3333, x_3^* = 0)$ and $f_3^* = 0$ with $x_1^* = x_2^* = x_3^* = 0$.

The solution by the proposed algorithm:

To solve such problem, it makes sense to begin by genetic algorithm. It selects at first some values for its parameters as seen in the following table.

The first obtained solution from GA is as follows:

$x_1 = 2.5080$, $x_2 = 3.1743$, $x_3 = 2.9268$. Thus, $f_1 = -24.932$, $f_2 = -13.209$, $f_3 = 18.019$, total OFs = -20.122, and the distance = 22.894.

Table 2. Parameters of genetic algorithm for example (3.3)

Parameters	Range/ Value/ Function
Population initial range	[0.0; 3.3333], [0.0; 3.2880], [0.0; 3.2412], [0.0; 3.2069]
Population size	10, 6, 18
Crossover fraction	0.5, 0.4, 0.6, 0.9
Mutation function	Uniform
Fitness limit	-31.5
Stall generation limit	Infinity
Stall time limit	Infinity
Penalty parameter	0.1

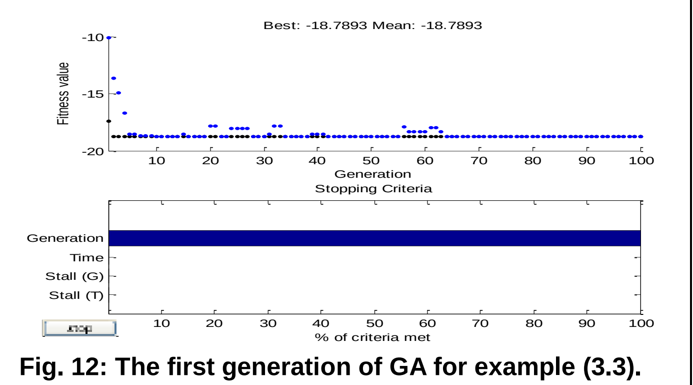


Fig. 12: The first generation of GA for example (3.3).

After some iterations, the last solution of GA is as follows: $x_1 = 0.0985$, $x_2 = 2.8460$, $x3 = 0.1764$. Then, $f_1 = -8.140$, $f_2 = -12.238$, $f_3 = 8.473$, total OFs = -11.905, $C_1 = -7.7338$, $C_2 = 8.734$, distance = 9.7785. **Fig. 12** shows the fitness function for last generation of this example following an increase in **Fig. 11** by 4.7962 but a decrease in the initial point from [0.0: 3.3333] to [0.0: 3.2069] with some changing in the described parameter values. Also, the distance decreased from 22.894 to 9.7785.

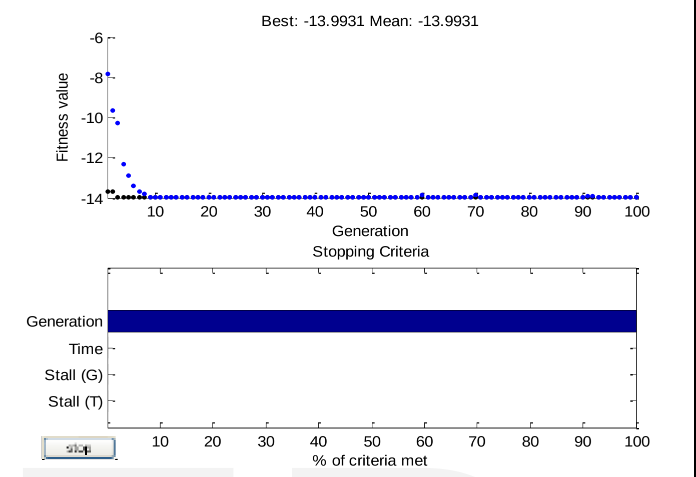


Fig. 13: The last generation of GA for example (3.3).

The second sub algorithm (Bacterial) starts with re-initialization of the outputs of GA as shown in the following table.

Table 3. Controller parameters of the BFA used in example (3.3)

Data	Range/ Value
The initial solution	**At first generation:** p_1= [0.0000; 0.0000; 0.0000; 2.8916; 3.0000; 3.2992; 3.8005; 3.8392] p_2= [0.0000; 2.6667; 3.3333; 3.3472; 3.3531; 3.3668; 3.6676; 4.0000] p_3= [0.0000; 0.0000; 0.0311; 2.0000; 3.2000; 3.2163; 3.2527; 3.7663] **At second generation:** p_1= [0.0000; 0.0000; 0.0000; 0.0000] p_2= [3.3333; 2.6668; 2.6667; 0.0000] p_3= [0.0311; 0.0002; 0.0000; 0.0000] . . **At last generation:** p_1= [0.0000; 0.0000] p_2= [2.7223; **2.7218**] p_3= [0.0000; 0.0000]
S	8, 4, 2
Nc	1, 2
Ns	2, 1
Nre	1, 2
Ned	4, 1
Ped	0.001, 0.01, 0.1
The run length	0.0001*ones(s,1), 0.01*ones(s,1), 0.001*ones(s,1)

The first solution of BFO is as follows:

x_1= 0.0003, x_2= 2.6668, x_3= 0.0002, f_1 = -7.112, f_2 = -12.110, f_3 = 7.112, total OFs = -12.110, C_1 = 13.9993, C_2 = 1.9994, C_3 = 2.8176e-004, C_4 = 2.6668, C_5 = 2.1643e-004, and distance = 9.088.

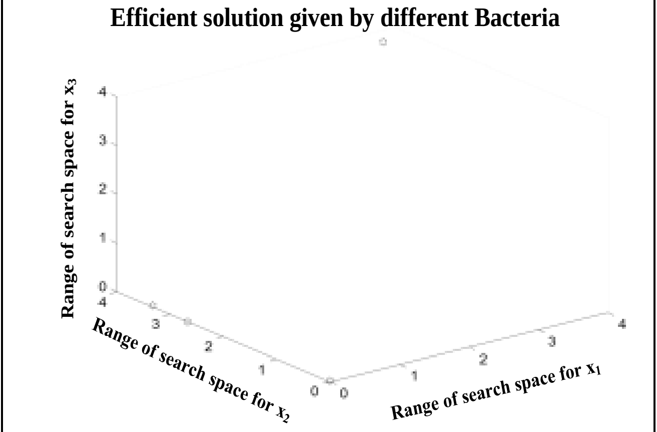


Fig. 14: The first solutions of BFO generations for example (3.3).

After several generations the final results are:
x_1= 0.0, x_2= 2.7220, x_3= 0, f_1 = -7.4093, f_2 = -12.4093, f_3 = 7.4093, total OFs = -12.4093, C_1 = 14.1660, C_2 = 1.8340, and distance = 9.0721.
Again, the BFO is successful in solving the problem. The obtained results show that the distance from the utopia point is considerably reduced over the GA. The following figure illustrates the final best positions of the bacteria. In summary, good performance (lower distance from the utopia point and acceptable settling time) is obtained.

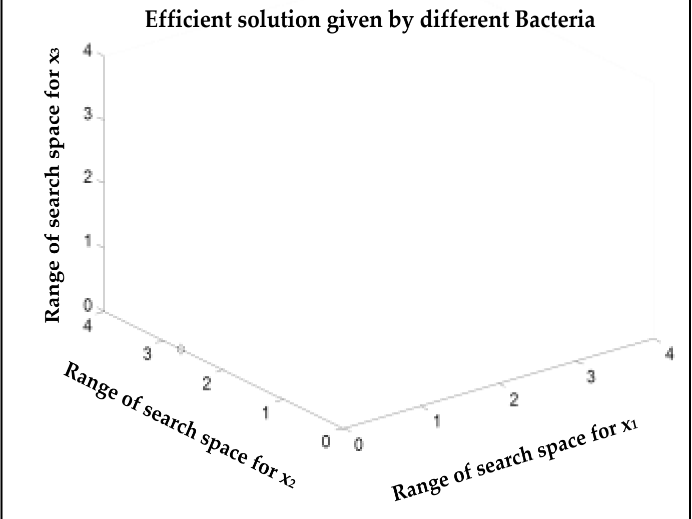


Fig. 15: The last solutions of BFO for example (3.3).

To continue with harmony algorithm, at first, the initial values of its parameters are setting as seen in the **Table 4**.
The first solutions of HS (*at best value and worst value in the iteration*) are:
x_1= 0, x_2= 2.6664, x_3= 0, f_1 = -7.1097, f_2 = -12.1097, f_3 = 7.1097, total of objectives = -12.1097, C_1 = 13.9992, C_2 = 2.0008, and distance = 9.0868.

Table 4. Parameters of the HSA for example (3.3)

Data	Range/ Value
Initial solution	**At first iteration:** [0.0000 0.0000 0.0000 2.8916 3.0000 3.2992 3.8005 3.8392; 0.0000 2.6667 3.3333 3.3472 3.3531 3.3668 3.6676 4.0000; 0.0000 0.0000 0.0311 2.0000 3.2000 3.2163 3.2527 3.7663] **At second iteration:** [0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000; 2.6667 3.3333 3.3472 3.3531 3.3668 3.6676 4.0000; 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000] . . **At last iteration:** [0.0000 0.0000 0.0000 0.0000 0.0000; **2.7217** 2.7224 2.7474 3.3332 3.3333; 0.0000 0.0000 0.0000 0.0000 0.0000]
NI	7000
HMS	4, 2
HMCR	0.2, 0.7
PAR min	0.7, 0.4
PAR max	0.9
bw min	0.5, 0.1
bw max	0.6, 0.7
Penalty parameter	-0.0001

The last values at best and worst generation of HS are:
x_1= 0, x_2= 2.7220, x_3= 0, f_1 = -7.4093, f_2 = -12.4093, f_3 = 7.4093, total of objectives = -12.4093, C_1 = 14.166, C_2 = 1.834, C_3 = 0, C_4 = 2.7220, C_5 = 0, and distance = 9.0721.
It is easy to see that these results from both BFO and HS are equivalent but with less computational time is achieved by HS.
On other hand, one of the Pareto-optimal solutions in [7] is **x**

= (0.00, 0.05, 0.99) that has distance from the utopia point = 17.3796. So, the above results indicate that the hybrid algorithms are superior to the others. They consume considerably short search time.

Example (3.4):

The following benchmark problem was used in the paper [10]. This problem was relatively difficult becauce the constraints divided the Pareto frontier into five regions which created difficulties for optimization algorithms to find all parts of the Pareto frontier as shown in the figure **(18)**. It consists of two nonlinear objectives and six constraints.

Min: $f_1 = -\,(25(x_1-2)^2 + (x_2-2)^2 + (x_3-1)^2 + (x_4-4)^2 + (x_5-1)^2)$,

Min: $f_2 = (x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2)$,

Subject to: $x_1 + x_2 - 2 \geq 0$,

$-x_1 - x_2 + 6 \geq 0$,

$x_1 - x_2 + 2 \geq 0$,

$-x_1 + 3x_2 + 2 \geq 0$,

$-(x_3-3)^2 - x_4 + 4 \geq 0$,

$(x_5-3)^2 + x_6 - 4 \geq 0$,

$x_1, x_2, x_6 \in [0;10]$, $x_4 \in [0;6]$, $x_3, x_5 \in [1;5]$.

The first process of the proposed algorithm (Genetic search):

This section starts with showing the parametric set up of the algorithmic parameters. These parameters are given below. Population initial range**:** [0.0: 5.0], Population size: 12, Crossover fraction: 0.5, Mutation function: Uniform, Fitness limit: -148.0, Stall generation limit: Infinity, Stall time limit: Infinity, and Penalty parameter: 0.01.

After one generation, the obtained solutions are as follows: $x_1 = 4.5843$, $x_2 = 1.6521$, $x_3 = 0.3979$, $x_4 = 2.9167$, $x_5 = 0.5045$, $x_6 = 0.0207$, $f_1 = -168.8678$, $f_2 = 32.6657$, total of objectives = -136.2021, C_1 =4.2364, C_2 = -0.2364, C_3 =4.9322, C_4 =2.3720, C_5 = -5.6876, C_6 = 2.2482, and the distance from utopia point =35.4568. This process of GA is below graphed.

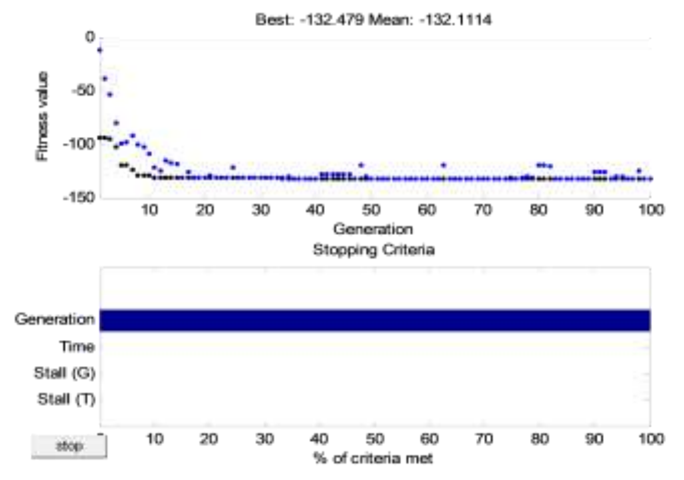


Fig. 16: The first generation of GA for example (3.4).

But after several iterations, the last solutions are:
x1 = 4.4095, x2 = 1.0231, x3 = 0.7322, x4 = 1.6200, x5 = 0.2534, x6 = 0.6058, f1 = -152.3901, f2 = 24.0821, total of objectives = -128.3080, C1 = 3.4326, C2 = 0.5674, C3 = 5.3864, C4 = 0.6598, C5 = -2. 6729, C6 = 4.1496 and distance = 20.5564.

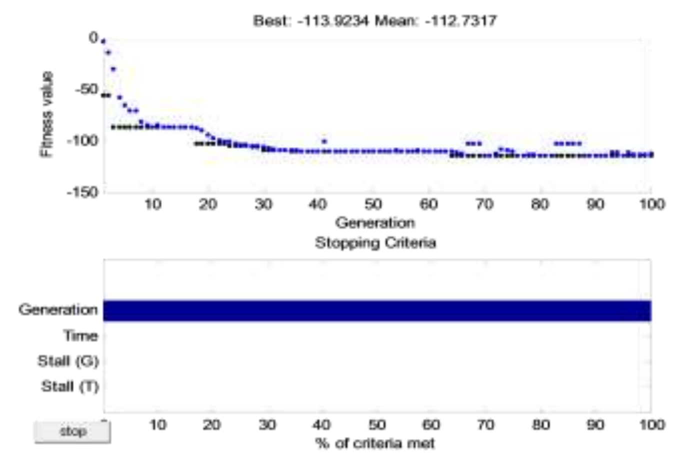


Fig. 17: The last generation of GA for example (3.4).

The second process of the proposed algorithm (*Bacterial foraging*):

To create the first generation, BFA sets some values for its controller parameters as shown in the following table.

Table 5. Controller parameters of the BFA used in example (3.4)

Data	Range/ Value

The initial solution	**At first generation:**
	p1= [0.0000; 1.0000; 4.2764; 4.3157; 4.3762; 4.3945; 4.4095; 4.4197; 4.4339; 4.4394; 4.4436; 4.4556; 4.4940; 4.5843]
	p2= [0.3154; 0.4392; 0.7484; 0.7584; 0.7631; 0.9234; 1.0000; 1.0231; 1.1211; 1.2841; 1.4701; 1.6521; 1.6763; 2.0000]
	p3= [0.1895; 0.2811; 0.2880; 0.3979; 0.6405; 0.7322; 1.0000; 1.0246; 1.0957; 1.1875; 1.2018; 1.7322; 2.0836; 5.0000]
	p4= [0.0000; 0.0000; 1.0090; 1.2875; 1.6200; 1.6993; 2.0557; 2.1389; 2.1951; 2.3347; 2.6410; 2.6511; 2.7221; 2.9167]
	p5= [0.2534; 0.3572; 0.4655; 0.4983; 0.5045; 0.5396; 0.5638; 0.6520; 0.7395; 0.8358; 1.0000; 1.2796; 1.9686; 5.0000]
	p6= [0.0000; 0.0000; 0.0033; 0.0098; 0.0207; 0.0977; 0.1564; 0.2762; 0.6058; 0.7952; 0.8240; 1.3017; 0.3194; 1.3471]
	
	At second generation:
	p1= **[4.4095; 4.3947; 4.3945; 4.3762; 4.3157; 4.2764]**
	p2= **[1.0000; 0.9235; 0.9234; 0.7631; 0.7584; 0.7484]**
	p3= **[1.2018; 1.1875; 1.0957; 1.0246; 1.0000; 0.7322]**
	p4= **[1.6994; 1.6200; 1.2875; 1.0090; 0.0000;**

	0.0000] p_5= **[5.0000; 1.9686; 1.2796; 1.0000; 0.8358; 0.5397]** p_6= **[0.0978; 0.0207; 0.0098; 0.0033; 0.0000; 0.0000]** . . . **At last generation:** p_1= [4.2777; 4.2777; 4.2777; 4.2777] p_2= [1.0000; 1.0000; 1.0000; 1.0000] p_3= [1.0000; 1.0000; 1.0000; 1.0000] p_4= [0.0000; 0.0000; 0.0000; 0.0000] p_5= **[1.0003; 1.0000; 0.9998; 0.9994]** p_6= [0.0000; 0.0000; 0.0000; 0.0000]
S	14, 6, 4
Nc	1, 2
Ns	2, 1
Nre	1
Ned	3, 1
Ped	0.01
The run length	0.0001*ones(S,1), 0.001*ones(S,1), 0.01*ones(S,1), 0.1*ones(S,1), 0.12*ones(S,1), 0.02*ones(S,1)

After several generations of BFO, the two sub-algorithms converge to the best result, which hopefully represents the favorable efficient solution of the problem to decision maker. The solutions that obtained by BFA are presented in the following table.

Table 6. The first and last solution of BFO for example (3.4)

Data	The first solution	The last solution
x_1	4.3947	4.2777
x_2	0.9235	1.0000
x_3	0.7322	1.0000
x_4	1.6994	0.0000
x_5	0.5397	1.0000
x_6	0.0978	0.0000
f_1	-150.0999	-146.6979
f_2	28.8912	21.2987
Total objectives	-126.2087	-125.3992
C_1	3.3182	3.2777
C_2	0.6818	0.7223
C_3	5.4712	5.2777
C_4	0.3758	0.7223
C_5	-2.8423	0.0000
C_6	2.1509	0.0000
Distance	20.0017	17.34765

After that, the harmony search works with the outputs of GA and all individual objective f_i^*; as initial solutions; to verify the GA-BFOA performance. Where, f_1^*= -148.0 with ($\mathbf{x_1}^*$= 0.0, $\mathbf{x_2}^*$= 2.0, $\mathbf{x_3}^*$= 5.0, $\mathbf{x_4}^*$= 0.0, $\mathbf{x_5}^*$= 5.0, $\mathbf{x_6}^*$= 8.0), and f_2^*= 4.0 with ($\mathbf{x_1}^*$= 1.0, $\mathbf{x_2}^*$= 1.0, $\mathbf{x_3}^*$= 1.0, $\mathbf{x_4}^*$= 0.0, $\mathbf{x_5}^*$= 1.0, $\mathbf{x_6}^*$= 0.0).

Table 7. Parameters of the HSA for example (3.4)

Data	Range/ Value
Initial solution	**At first iteration:** [0.0000 1.0000 4.2764 4.3157 4.3762 4.3945 4.4095 4.4197 4.4339 4.4394 4.4436 4.4556 4.4940 4.5843; 0.3154 0.4392 0.7484 0.7584 0.7631 0.9234 1.0000 1.0231 1.1211 1.2841 1.4701 1.6521 1.6763 2.0000; 0.1895 0.2811 0.2880 0.3979 0.6405 0.7322 1.0000 1.0246 1.0957 1.1875 1.2018 1.7322 2.0836 5.0000; 0.0000 0.0000 1.0090 1.2875 1.6200 1.6993 2.0557 2.1389 2.1951 2.3347 2.6410 2.6511 2.7221 2.9167; 0.2534 0.3572 0.4655 0.4983 0.5045 0.5396 0.5638 0.6520 0.7395 0.8358 1.0000 1.2796 1.9686 5.0000; 0.0000 0.0000 0.0033 0.0098 0.0207 0.0977 0.1564 0.2762 0.6058 0.7952 0.8240 1.3017 0.3194 1.3471] **At second iteration:** [4.2764 4.3157 4.3762 4.3945 4.4095 4.4197 4.4339 4.4394 4.4436 4.4556 4.4940 4.5843; 0.7484 0.7584 0.7631 0.9234 1.0000 1.0231 1.1211 1.2841 1.4701 1.6521 1.6763 2.0000; 0.2880 0.3979 0.6405 0.7322 1.0000 1.0246 1.0957 1.1875 1.2018 1.7322 2.0836 5.0000; 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000; 0.4655 0.4983 0.5045 0.5396 0.5638 0.6520 0.7395 0.8358 1.0000 1.2796 1.9686 5.0000; 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000] . . **At last iteration:** [4.2777 4.2777; 1.0000 1.0000; 1.0000 1.0000; 0.0000 0.0000; **0.9997** 1.0000; 0.0000 0.0000]
NI	1000, 2000
HMS	7, 5, 4

HMCR	0.9, 0.09, 0.7, 0.07
PAR min	0.4, 0.04
PAR max	0.9, 0.09
bw min	0.1, 0.01
bw max	0.7, 0.07
Penalty parameter	-0.0001

The first solution of HS with two ranges of iterations:
At best value in the iteration:
x_1=0.0002, x_2= 0.3156, x_3= 0.1897, x_4= 0.0000, x_5= 0.2534, x_6= 0.0000, f_1 = -120.0312, f_2 = 0.1998, total objectives = -119.8314, C_1 = -1.6842, C_2 = 5.6842, C_3 = 1.6846, C_4 = 2.9466, C_5 = -3.8978, C_6 = 3.5438, distance from utopia point = 28.2257.
At worst value in the iteration:
x_1= 0.0006, x_2= 0.3163, x_3= 0.1903, x_4= 0.0000, x_5= 0.2544, x_6= 0.0000, f_1 = -119.9864, f_2 =0.2010, total of objectives = -119.7854, C_1 = -1.6831, C_2 = 5.6831, C_3 = 1.6843, C_4 = 2.9483, C_5 = -3.8944, C_6 = 3.5383, distance = 28.2700.
The final solution:
At best value in the iteration:
x_1= x_2= x_3 = x_4= x_5= x_6= 0.0000, f_1 = -122.0000, f_2 = 0.0000, C_1 = -2.0000, C_2 = 6.0000, C_3 = 2.0000, C_4 = 2.0000, C_5 = -5.0000, C_6 = 5.0000, distance = 26.3059.
At worst value in the iteration:
x_1= 4.2777, x_2= 1.0000, x_3= 1.0000, x_4= 0.0000, x_5= 1.0000, x_6= 0.0000, f_1 = -146.6979, f_2 = 21.2987, total of objectives = -125.3992, C_1 = 3.2777, C_2 = 0.7223, C_3 = 5.2777, C_4 = 0.7223, C_5 = 0.0000, C_6 = 0.0000, distance = 17.34765.

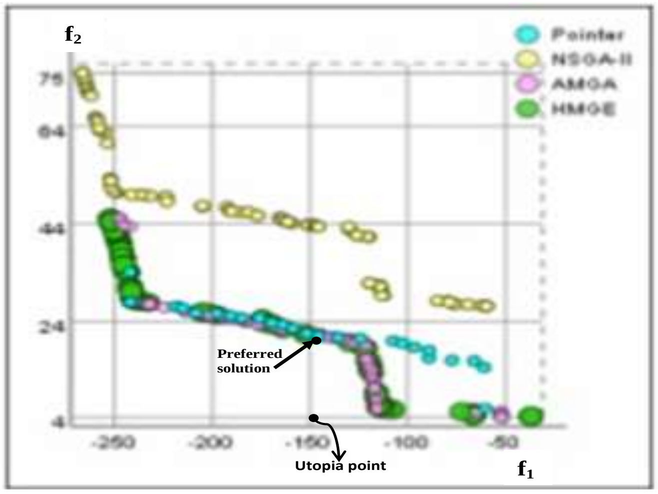


Fig. 18: The Pareto-optimal and Preferred solution of example (3.4).

As is illustrated in the above results and figure, the proposed algorithm has faster convergence compared with similar algorithms. Also, the sub-algorithm BFO and harmony search produce perfect results than other sub-algorithm (GA) for all types of such variant problems.

4 EXPERIMENTAL RESULTS and ANALYSES

This paper presents a new approach by hybridizing genetic algorithm with bacterial foraging optimization and harmony search. The different comparative examples are examined to show the effectiveness of the proposed algorithm. The following performance measures are used for this comparative study: (a) quality of the final solution based on achieving minimum distance from the utopia point, (b) speed of convergence towards the favorite efficient solution, and (c) scalability of the algorithms against the growth of dimensions for different problems.

The experiments were carried out to compare three intelligent algorithms that are included in the proposed algorithm on four test problems with dimensions of one, two, three and six. ***The major findings of this study are as follows:***

- Despite the genetic optimization technique requires a very long run time that may be several hours or more several hours, it gave good results (as in the first and second examples) depending on the size of the system under study.
- BFO algorithm has been successfully applied in several domains of the most complex problems over the GA.
- Although the dimensions of test problems are increased, BFO and HS exhibit good search capabilities and convergence faster than GA algorithm on most of the test problems.
- The experimental results suggest that the results from HS are better than results from other models. The performance of BFO is satisfactory for all the problems. Where, the performance of BFA is nearby same as HS.
- Also, this comparative study shows the superiority of the proposed algorithm over other algorithms commonly used in the literature of the multi-objective optimization problems.
- The harmony algorithm works very well in different such problems. Where, it yielded more accurate final results consuming lesser amount of computational time in all the test cases. So, this is a promising technique that can be used in complex problems.
- The novel approach will be useful to solve many different of multi-objective applications.

5 CONCLUSION

The results from the proposed algorithm are shown to be robust and extendable, suggesting the potential of applying the BFO and HS for harder higher-dimensional and dynamic optimization problems in different applications.

HS has many features that make it as a preferable technique and also to be combined with other meta-heuristic algorithms.

ACKNOWLEDGMENT

- **At first thanks be to ALLAH** for his guidance and supports in showing us the path.
- **Secondly**, I would like to thank Prof. Dr. Mahmoud Mostafa El-Sherbiny (*Operations Research and Management Department, Institute of Statistical Studies and Research, Cairo University, Egypt*) for his valuable comments.

REFERENCES

[1] Alia Youssef Gebreel, "An overview of genetic algorithm, bacterial foraging algorithm, and harmony search algorithm", *Global Scientfic Journals*, Vol. 6, No.9, PP. 165-189, Septemer (2018).

[2] Indresh Kumar Gupta, Jeetendra Kumar, "VEGA and MOGA an approach to multi-objective optimization", *International Journal of Advanced Research in Computer Science and Software Engineering*, ISSN: 2277 128X, Vol. 5, No. 4, PP. 865- 870, (2015).

[3] ***Internet:*** Antonio L´opez Jaimes, Sa´ul Zapotecas Mart´ınez, Carlos A. Coello Coello, "An introduction to multiobjective optimization techniques", Nova Science Publishers, Inc., Chapter (1), PP. 1-26.

[4] ***Internet:*** Performing a multiobjective optimization using the genetic algorithm, https:// www. mathworks. com/ help/ gads/ examples/ performing- a- multiobjective- optimization- using- the- genetic-algorithm.html.

[5] Kalyanmoy Deb, J. Sundar, Udaya Bhaskara Rao N. and Shamik Chaudhuri, "Reference point based multi-objective optimization using evolutionary algorithms", *International Journal of Computational Intelligence Research*, ISSN 0973-1873 Vol.2, No.3, PP. 273–286, (2006).

[6] Kang Seok Lee, Zong Woo Geem, "A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice", *Computer methods in applied mechanics engineering*, 194, PP. 3902–3933, (2005).

[7] Majid Rafei, Samin Ebrahim Sorkhabi, Mohammad Reza Mosavi, "Multi-objective optimization by means of multi-dimensional MLP neural networks", *Neural Network World* 1/14, PP. 31- 56, (2013).

[8] MATLAB software, Version 7.0, (2004).

[9] Rasleen Jakhar, Navdeep Kaur, and Ramandeep Singh, "Face recognition using bacteria foraging optimization-based selected features", (IJACSA) *International Journal of Advanced Computer Science and Applications*, Special Issue on Artificial Intelligence, PP. 106-111.

[10] Valdimir Sevasty Anov, "Hybrid multi- gradient explorer algorithm for global multi-objective optimization", American Institute of Aeronautics and Astronautics, (10), 94-99, PP. 1-15, eartius, Inc., (2013).

[11] Waiel Fathi Abd El-Wahed, "A seminar on intelligent optimization", Faculty of Computers & Information, Menoufia University, Egypt, October, (2010).

[12] W. J. Tang and Q. H. Wu, "Biologically inspired optimization: a review", *Transactions of the Institute of Measurement and Control* Vol. 31, No. 6, PP. 495–515, (2009).