Global Scientific JOURNALS

GSJ: Volume 6, Issue 9, September 2018, Online: ISSN 2320-9186
www.globalscientificjournal.com

An Overview of Genetic Algorithm, Bacterial Foraging Algorithm, and Harmony Search Algorithm

Alia Youssef Gebreel

(Ph.D. Degree in Operations Research from Institute of Statistical Studies and Research (ISSR), Cairo University, Egypt).

ABSTRACT

This paper presents a brief background and architecture needed to recognize, formulate, and solve the multi-objective optimization problems by genetic algorithm, bacterial foraging algorithm and harmony algorithm. A comparison of these three algorithms for this class of problems is also included. Penalty function method for handling constrained problems is described. Finally, the conclusion is reported.

KeyWords

Multi-objective Optimization Problems, Genetic Algorithm (GA), Bacterial Foraging Algorithm (BFA), Harmony Search Algorithm (HSA).

1. Introduction

The *exact methods* are usually the deterministic methods that search the search space exhaustively and identify the Pareto–optimal set. While the size of the solution space is growing, the problem becomes more complex. Thus, these methods become very much expensive and less applicable. So, the heuristic optimization methods are applied. Heuristic is a computational method that uses iterations to improve an optimal solution of any problem. This method makes very less assumptions about the problem being optimized and can search very large spaces of candidate solutions. Many heuristics implement some form of stochastic optimization. Due to this reason, these methods are also called as combinatorial methods. On the other hand, *Meta-heuristics* can be considered as a generic technique which can be used to solve problems of different domains. *Heuristics* are often problem specific, but *meta-heuristics* are quite generic and hence more powerful. Examples of meta–heuristics include genetic algorithm (GA), neural network (NN), tabu search (TS), simulated annealing (SA), ant colony optimization (ACO), particle swarm optimization (PSO), Bacterial foraging algorithm (BFA), harmony search (HS), etc. [24, 36].

Many real world problems have several criteria with a group of constraints to be optimized. The resulted solutions are called non-dominated, efficient, or Pareto- optimal solutions. Artificial intelligent optimization algorithms have been one of the most popular methods to find multiple trade-off solutions in single simulation. But when decision making is emphasized, the goal of solving a multi-objective optimization problem is referred to supporting a decision maker in finding the most preferred Pareto optimal solution according to his/her subjective preferences. So, the interactive methods use the preference information progressively during the optimization process. The main aspect in these approaches is that the decision maker provides some information during the optimization process from time to time. Such information is the direction of search, weight vector, reference points, and other factors. Since the decision maker is familiar to the optimization process, these techniques become popular in practice. Generally in an interactive method, the decision maker works together with analyst or an interactive computer program [19, 28, 29].

The methods for solving the multi- objective optimization problems can be classified according to information mode as follows:

a) Methods with a priori information:

Decision maker provides global preference information (weights, utility, goal values . . .). Analyst solves a single objective problem.

b) Methods with progressive information—interactive methods:

Decision maker provides local preference information. Analyst solves local problems and provides current solutions.

c) Methods with a posteriori information:

Analyst provides a non-dominated set. Decision maker provides global preference information on the non-dominated set. Analyst solves a single objective problem.

There are proposed many methods from these categories. Most of the methods are based on trade-offs [43].

There are some earlier works that are related to this research. For example, Jian and Pratyush [26] designed goal programming model for capturing the decision maker's preference information. This model supports the search for the best compromise solutions in multi-objective optimization. The interactive step trade-off method is employed to generate a typical subset of efficient solutions of a multi- objective problem. It identifies and eliminates the possible inconsistent information which may exit in the preferences of decision maker. Also, it provides various ways to carry out post-optimality analysis for testing the robustness of the obtained best compromise solutions. Another approach worth noting is that of Hong, Zhigang and Ming [9]. They constructed the designer's pref-

erence structure model using artificial neural networks with the model parameter vectors as the input and the preference information. The desired output articulates by the designer over representative samples from the Pareto frontier. Then, the optimization problem is solved to search for improved solutions. This model is very effective in capturing different kinds of preference structures of the designer, and it can obtain the Pareto solution that satisfies the designer's preference. In addition, John and Prospero [27] suggested an approach to neural network training through the simultaneous optimization of architectures and weights with a Particle Swarm Optimization-based multi-objective algorithm. It dynamically generates a set of near-optimal feed-forward neural networks with their corresponding connection weights. This approach provided an effective means for training neural networks that is competitive with other evolutionary computation-based methods. But in the research of Javier, Miguel and Xavier [25], the user's information about his/ her preferences is taken into account within the search process. The preference functions convert these objective preferences into numbers. Problems found due to the multimodality nature of a generated single cost index are managed with genetic algorithm. It presented with examples showing its application in engineering design problems. Furthermore, Roberto, Fellow, IEEE, and Andrea [46] presented a preference-based evolutionary multi-objective optimization algorithm characterized by its ability to learn an arbitrary utility function from a decision maker who expresses preferences between couples of selected solutions. They point out that this work has ability to function without any *a priori* assumptions on the shape of the decision maker's utility function. It is a robust work as it can potentially withstand incomplete, imprecise, and even contradictory feedback by the decision maker. The alternation of evolutionary optimization and decision maker ranking can be organized according to a flexible schedule, depending on the willingness by the decision maker to interact more times during the solution process and by intrinsic characteristics of the problem to be solved and the complexity of the user preferences. The presented experimental results demonstrate the feasibility and effectiveness of the proposed algorithm on a variety of benchmark tasks, with both linear and non-linear user preferences. Ganesan, Shanmuga, and Kerana [8] developed a scientific methodology in determinant variable weights of multi-objectives. The roles of weights in a creative multi-objective decision-making or machine-learning of square measure analyzed, and therefore the weights square measure determined with the help of a standard neural network. Finally, Alia [1] introduced an adaptive interactive approach based on decision neural network. The neural network is used to capture and represent the decision maker's preference as the input to search for the desirable solution. Then by applying genetic algorithm, the most desirable solution is provided. The results show that the combination of neural network model with genetic algorithm gives a clear improvement in solving the interactive multi- objective optimization problems.

The primary aim of this search is to provide a general overview of the genetic algorithm, bacterial foraging optimization and harmony search that are implemented for optimization problems in the research using MATLAB software.

The rest of the paper is consisted of seven sections. These sections present a general description of GA, BFA and HS optimization model, and their adaptation to the problem. Also, a comparison of them is introduced that followed by penalty function method. Conclusions are finished the paper.

2. Genetic Algorithm

A genetic algorithm is a heuristic search that mimics the process of natural evolution concept coming from Darwin's theory of evolution. GA is started with a set of solutions (chromosomes) called population.

Fig. (1): Gene stairs

The solutions from one population are taken and used to form a new population. This is motivated by a hope; that the new population will be better than the old one. The selected solutions to form new solution (offspring) are choosen based on their fitness. This process is repeated until some condition (such as maximum number of generations or improvement of the best solution) is satisfied.

GA enables to solve unconstrained, bound-constrained, and general optimization problems, and it does not require the functions to be differentiable or continuous [16].

There are five phases namely initial population, fitness function, selection, crossover, and mutation. Where, selection, crossover, and mutation are called reproduction operators. The flow chart of GA is presented in the following figure.

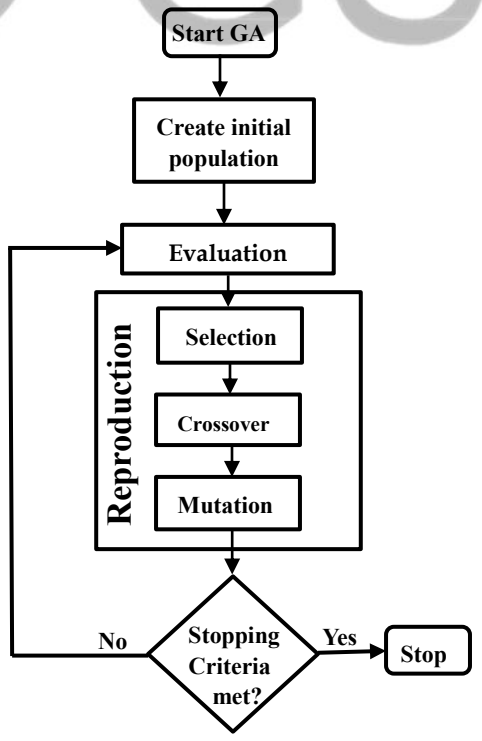


Fig. (2): Steps of a simple genetic algorithm.

2.1 Initial population:

It begins with randomly generated population of ***n*** chromosomes (that are satisfactory to the problem). There are two major ways to represent a chromosome:

- **Binary representation:**

  As shown in the following figure, each decision variable is represented in more than one gene.

Decision variable	X_1			X_2			X_3		
Genotype	0	0	1	1	1	0	1	0	1
phenotype		**1**			**6**			**5**	

Fig. (3): Binary representation.

This means that: X_1 =1, X_2 = 6, and X_3 = 5.

- **Floating point representation:**

  In this way, each decision variable is represented in only one gene as shown in the following figure.

X_1	X_2	X_3	… … … … … …	X_n
1	5	4	… … … … … …	2

Fig. (4): Floating point representation.

This means that: X_1 =1, X_2 = 5, X_3 = 4, … … ..., X_n =2.

There are two primary methods to initialize a population in a GA. They are:

- **Random Initialization:** Populate the initial population with completely random solutions.
- **Heuristic initialization:** Populate the initial population using a known heuristic for the problem.

Where, the GA terminologies are as follows:

- **Population;** that is a collection of individuals.
- **Individuals;** (genotypes, structures, phenotype) in a population quite often these individuals are called strings or chromosomes.
- **Chromosome;** a chromosome representation is needed to describe each individual in the population of interest for any GA. It is a collection of primitive features called genes. Genes are features, characters, or decoders [6, 15, 17, 18].
- **Gene;** it is a single feature within a chromosome. It may take on any of several values called alleles.
- **Allele;** it is a particular value that may be taken on by a gene. In general, different genes will have different alleles. String is made up of series of characters "genes".
- **Genotype;** it is the explicit genetic structure of a chromosome.
- **Phenotype;** this is a physical expression of the genotype "chromosome".

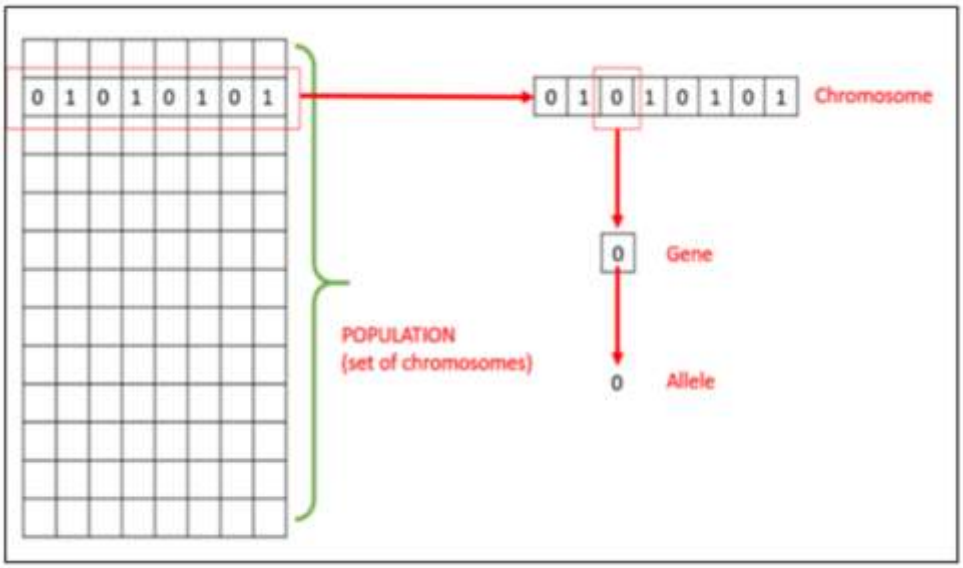


Fig. (5): Diagram of a population, chromosome, gene, and allele.

- **Decoding and Encoding;** Decoding is a process of transforming a solution from the genotype to the phenotype space, while encoding is a process of transforming from the phenotype to genotype space.

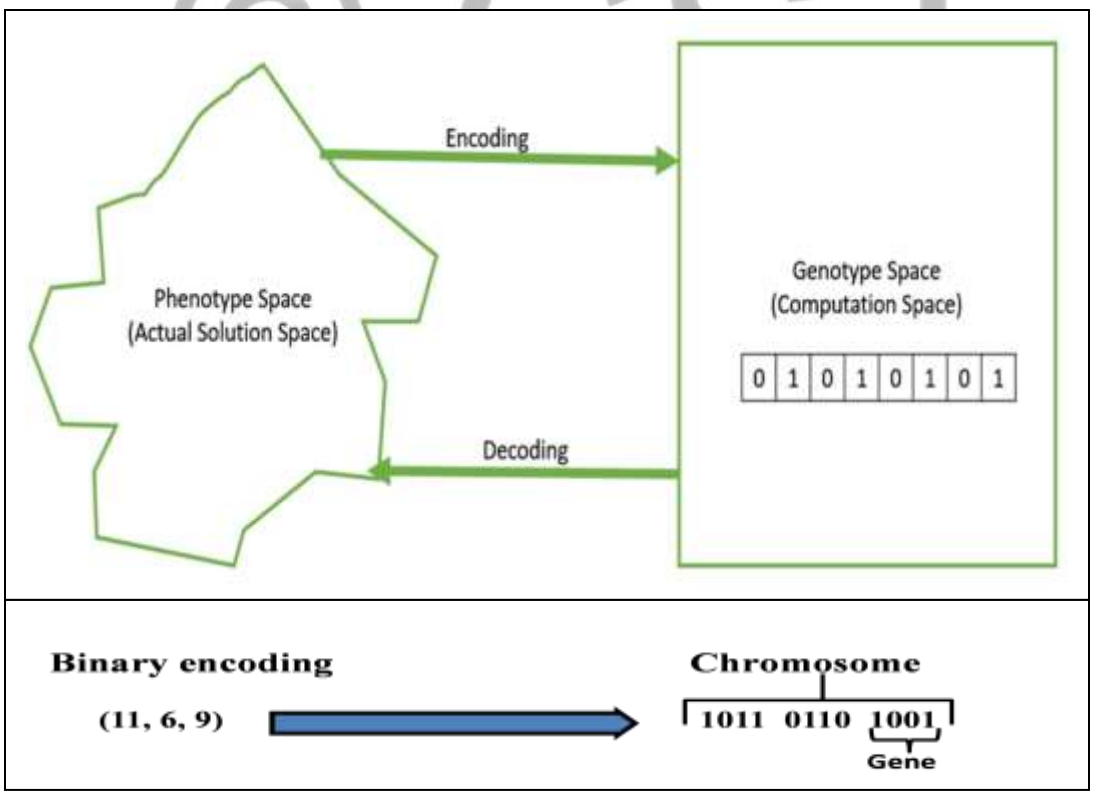


Fig. (6): Decoding and encoding processing.

2.2 Fitness function:

The fitness function produces the next generation of population. It is a measure of how will adapt the specific individual. A good fitness function should return better chromosomes. The fitness function gives a score to each chromosome. The probability of being chosen for reproduction is based on decision maker's fitness score. A fitness function value quantifies the optimality of a solution. The value is used to rank a particular solution against all the other solutions.

2.3 Selection:

Two pairs are selected at random to reproduce. They are selected based on their fitness function score. Fitness may be determined by an objective function or by a subjective judgment. The primary objective of the selection operator is to emphasize the good solutions and eliminate the bad solutions in a population while keeping the population size constant.

There are different techniques to implement selection in genetic algorithms:

- **Tournament selection;** in which a group of parents is selected and a tournament is held to decide which of the individuals will be the parent.

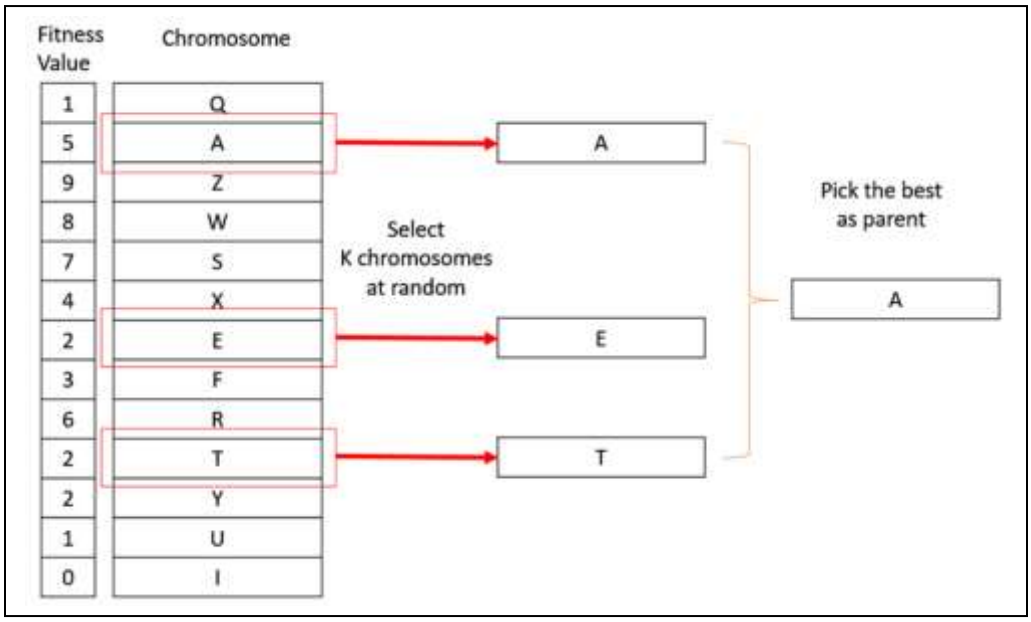


Fig. (7): A graphical representation of tournament selection.

Tournament Selection is also extremely popular in literature as it can even work with negative fitness values.

- **Roulette wheel selection;** that chooses parents by simulating a roulette wheel, where each individual is given a chance to become a parent in proportion to its fitness evaluation.

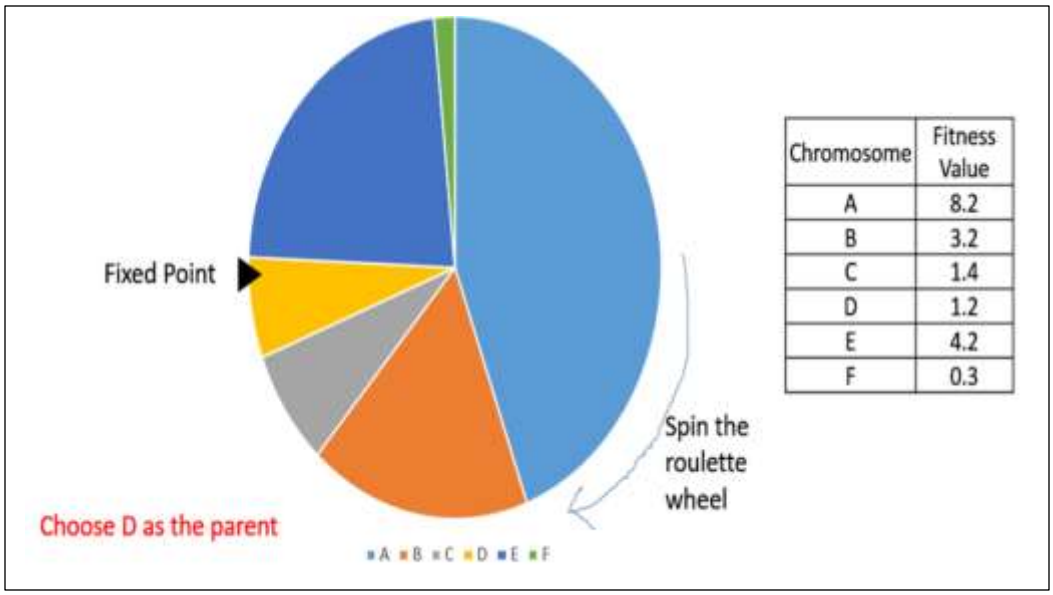
The following steps can be used to implement the roulette wheel selection:

1- Calculate Sum = the sum of a finesses.

2- Generate a random number between 0 and Sum.

3- Starting from the top of the population, keep adding the finesses to the partial sum (PS), till PS < Sum.

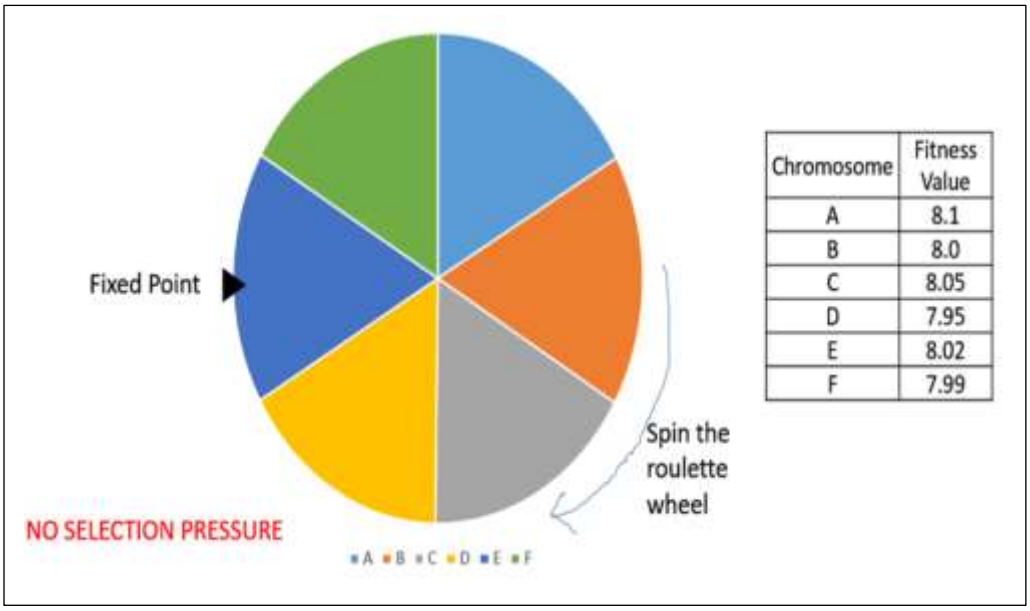
4- The individual for which (PS) exceeds (Sum) is the chosen individual.



Chromosome	Fitness Value
A	8.2
B	3.2
C	1.4
D	1.2
E	4.2
F	0.3

Fig. (8): A graphical representation of roulette wheel selection.

- **Proportionate selection;** that makes the scaled value of an individual proportional to its raw fitness score. It is to be noted that fitness proportionate selection methods don't work for cases where the fitness can take a negative value.
- **Uniform selection;** that chooses parents using the expectations and number of parents. Uniform selection is useful for de-bugging and testing, but is not a very effective search strategy.
- **Fitness ranking selection;** where individuals are sorted in order of raw fitness and given ranks. Good ranked individuals are chosen as parents. Rank Selection also works with negative fitness values and it is mostly used when the individuals in the population have very close fitness values (this happens usually at the end of the run).
  The higher ranked individuals are preferred more than the lower ranked ones.



Chromosome	Fitness Value
A	8.1
B	8.0
C	8.05
D	7.95
E	8.02
F	7.99

Fig. (9): A graphical representation ranking selection.

- **Remainder selection;** that assigns parents deterministically from the integer part of each individual's scaled value and then uses roulette selection on the remaining fractional part. For example, if the scaled value of an individual is 2.3, that individual is listed twice as a parent because the integer part is 2. After parents have been assigned according to the integer parts of the scaled values, the rest of the parents are chosen stochastically. The probability that a parent is chosen in this step is proportional to the fractional part of its scaled value [14, 38, 44, 47].

2.4 Crossover:

With a crossover probability, offspring are created by exchanges between the parents at the crossover point. Crossover has the greatest effect in improving the solution step- by- step. But, it may fall in local optimum solution. Some crossover methods can be useful or even necessary; which can be written down as follows:

- ***One-point crossover*** is a simple and often-used method for GAs which operates on binary strings as shown in the following figure.

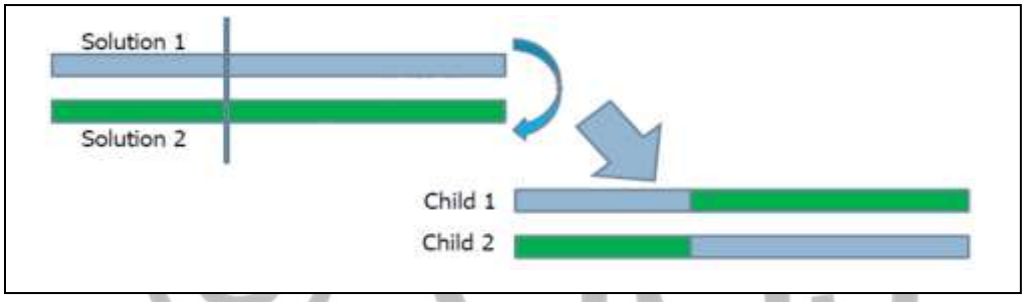


Fig. (10): A graphical representation of one- point crossover.

- ***N-point crossover:*** Instead of only one, ***N*** breaking points are chosen randomly. Every second section is swapped. Among this class, ***two-point crossover*** is particularly important.

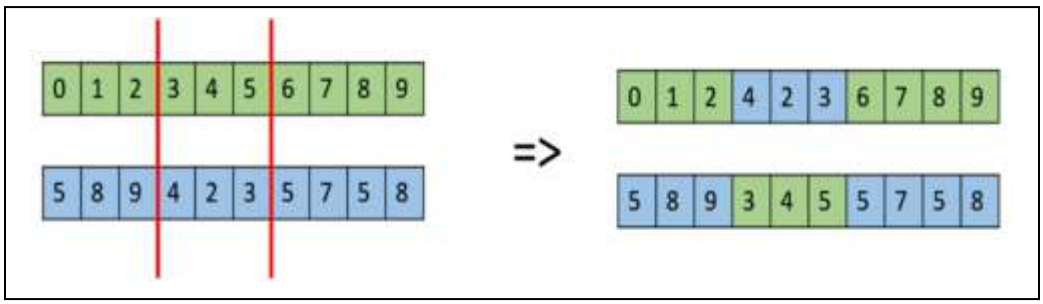


Fig. (11): A graphical representation of multi- point crossover.

- ***Segmented crossover:*** Similar to N-point crossover with the difference that the number of breaking points can vary.
- ***Uniform crossover:*** For each position, it is decided randomly if the positions are swapped.

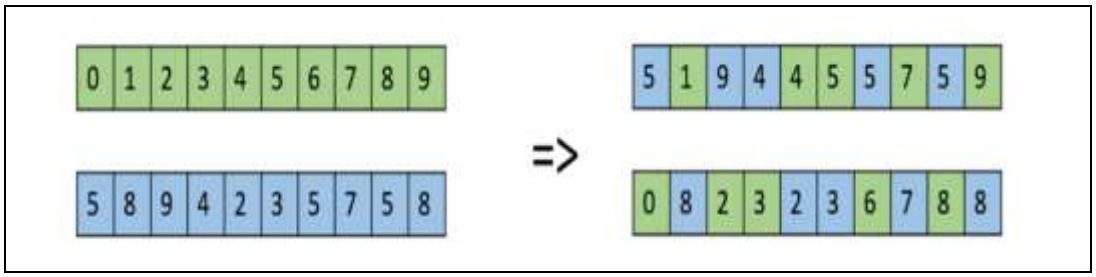


Fig. (12): A graphical representation of uniform crossover.

- ***Shuffle crossover:*** First a randomly chosen permutation is applied to the two parents, then N-point crossover is applied to the shuffled parents, finally, the shuffled children are transformed back with the inverse permutation.
- ***Intermediate crossover:*** it creates children by taking a weighted average of the parents.

2.5 Mutation:

With a mutation probability mutate new offspring at each position in chromosome. Similar to the case of crossover, the choice of the appropriate mutation technique depends on the coding and the problem itself. Mutation has the greatest effect in treating locality of crossover. It gets random solution that may lead GA to the global optimum.

Fig. (13): A graphical representation of a mutation.

A few alternatives can be summarized as follows:

- **Inversion of single bits:** With probability p_M, one randomly chosen bit is negated.
- **Bitwise inversion:** The whole string is inverted bit by bit with prob. p_M.

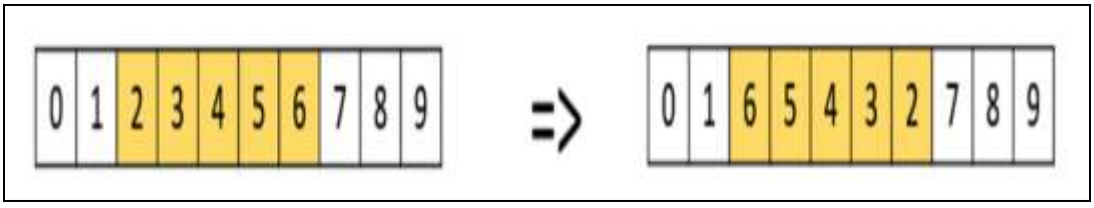


Fig. (14): A graphical representation of inversion mutation.

- **Random selection:** With probability p_M, the string is replaced by a randomly chosen one [17, 37, 48].
- **Gaussian mutation:** It adds a random number taken from a Gaussian distribution with mean 0 to each entry of the parent vector.

- **Uniform mutation:** It is a two-step process. First, the algorithm selects a fraction of the vector entries of an individual for mutation, where each entry has a probability rate of being mutated. In the second step, the algorithm replaces each selected entry by a random number selected uniformly from the range for that entry.

2.6 Genetic algorithm parameters:

The performance of the algorithm is influenced by different parameters like population size that represents number of individuals in each generation, crossover probability which is probability of creating new individuals via selected crossover type of selected individuals, number of generations represents the number of iterations of whole algorithm, mutation probability which is probability of creating new individuals via mutation based on selected individual and stopping criteria which may be specified by the user it may be number of generations or if a satisfactory solution found. Also, reproduction operators are Selection, Crossover, and Mutation. Selection means that two chromosome be selected from the population and crossover or mutation is applied [11, 47].

2.7 Stopping criteria options:

Stopping criteria determine what causes the algorithm to terminate. It can be specify the following options:

Generations: Specifies the maximum number of iterations the genetic algorithm will perform.

Time limit: Specifies the maximum time in seconds the genetic algorithm runs before stopping.

Fitness limit: The algorithm stops if the best fitness value is less than or equal to the value of Fitness limit.

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

Stall time: The algorithm stops if there is no improvement in the best fitness value for an interval of time in seconds specified by Stall time.

The different phases of **GA** are visualized [11] in the following figure.

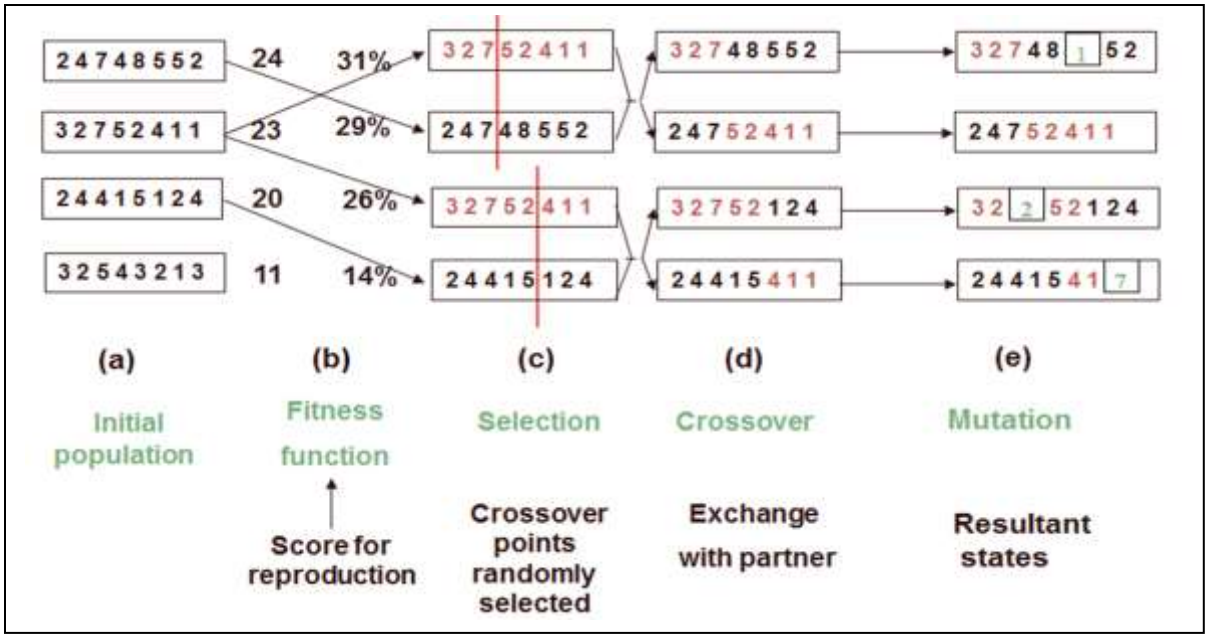


Fig. (15): A numerical example corresponds to the genetic phases.

2.8 Genetic algorithm in optimization problem:

Genetic algorithm is applied to optimize any problem according to the following steps:

1) The search space of all possible solutions is mapped onto a set of finite strings.

2) A set of solutions is selected (usually at random) and this constitutes the initial population.

3) Decoding process transforms from chromosome into decision variable.

4) Fitness is computed for each the individuals.

5) Encoding of solutions is as a vector string.

6) A set of individuals is selected to reproduce itself; the selection takes place in a random way.

7) From the selected set, applying different genetic operators generates some children. Genetic algorithm usually uses penalty function approach for constraint optimization problems to handle constraint (s). Where, it reduces the fitness of infeasible solutions, preferably so that the fitness is reduced in proportion with the number of constraints violated or the distance from the utopia point.

8) Some of the old individuals are eliminated from the population to allow the entrance of the new ones.

9) The new individuals are computed and they are included in the population. This step marks the end of a generation.

3. Bacterial Foraging Algorithm

Bacterial foraging algorithm (BFA) was originally proposed by Kevin M. Passino in 2000. It mimics the foraging behavior of Escherichia coli bacteria that live in human intestine [13, 24]. BF simulated the life cycles and the foraging behaviors of bacterial to solve the optimization problems [21, 32].

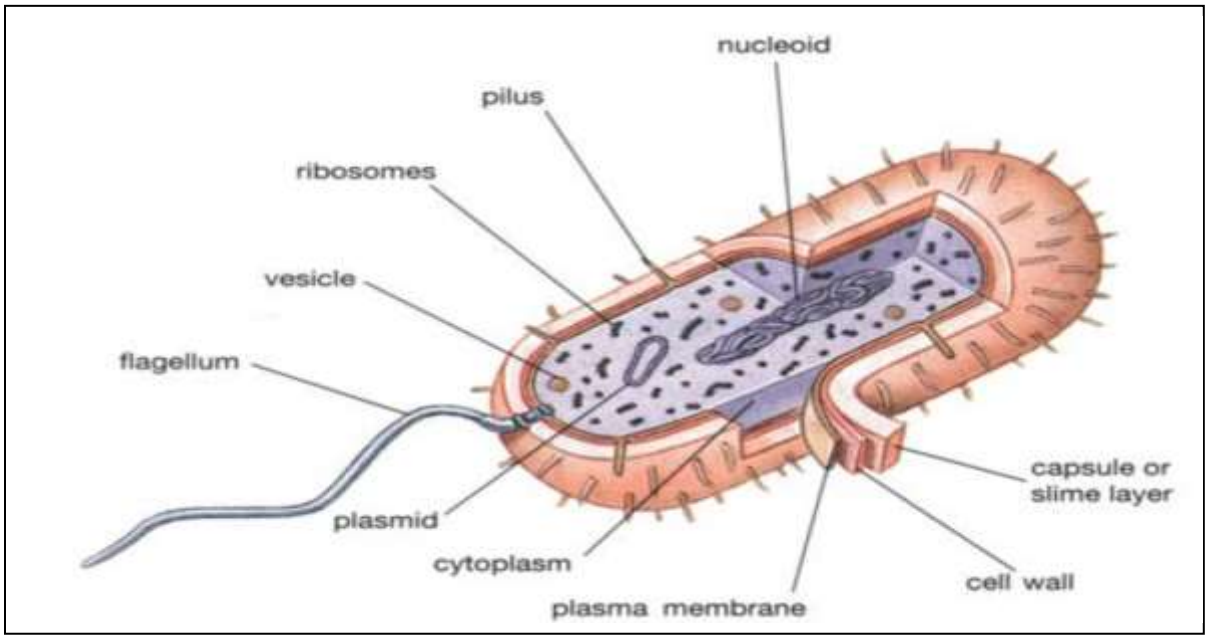


Fig. (16): Escherichia coli (*E. coli*) bacterium.

Bacteria perceive the direction to food based on the gradients of chemicals in their environment. Similarly, bacteria secrete attracting and repelling chemicals into the environment and can perceive each other in a similar way.

Using locomotion mechanisms (such as flagella) bacteria can move around in their environment, sometimes moving chaotically (tumbling and spinning), and other times moving in a directed manner that may be referred to as swimming. Bacterial cells are treated like agents in an environment, using their perception of food and other cells as motivation to move, and stochastic tumbling and swimming like movement to re locate. Depending on the cell-cell interactions, cells may swarm a food source, and/or may aggressively repel or ignore each other.

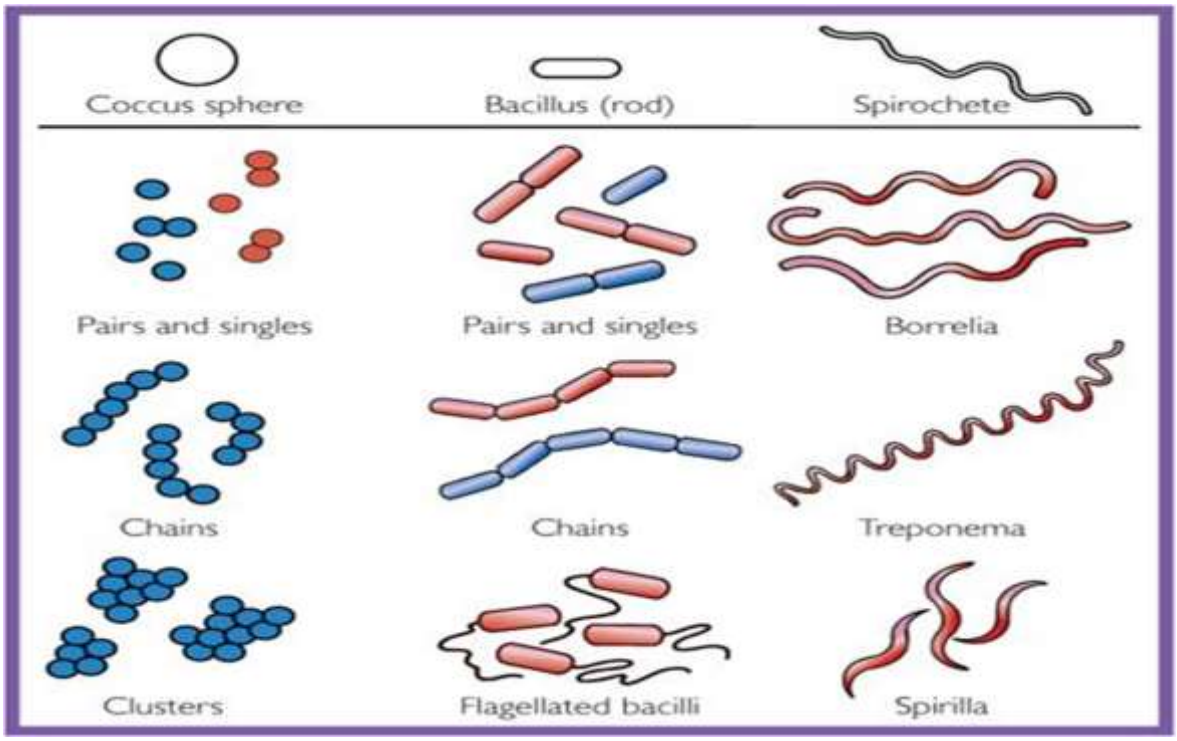


Fig. (17): Shapes of some bacteria.

The information processing strategy of the algorithm is to allow cells to stochastically and collectively swarm toward optima. This is achieved through a series of three processes on a population of simulated cells: chemotaxis, reproduction, and elimination-dispersal [5, 24].

The main steps of BFA are outlined below.

3.1 Chemotaxis:

This process simulates the movement of an E. coli cell through swimming and tumbling via flagella as seen in **Fig. (18)**, and **Fig. (19)**. Biologically an Escherichia coli (E. coli) bacterium can move in two different ways [12].

i) A unit walk with random direction represents a *Tumble.*

ii) A unit walk with the same direction in the last step indicates a *Run*.

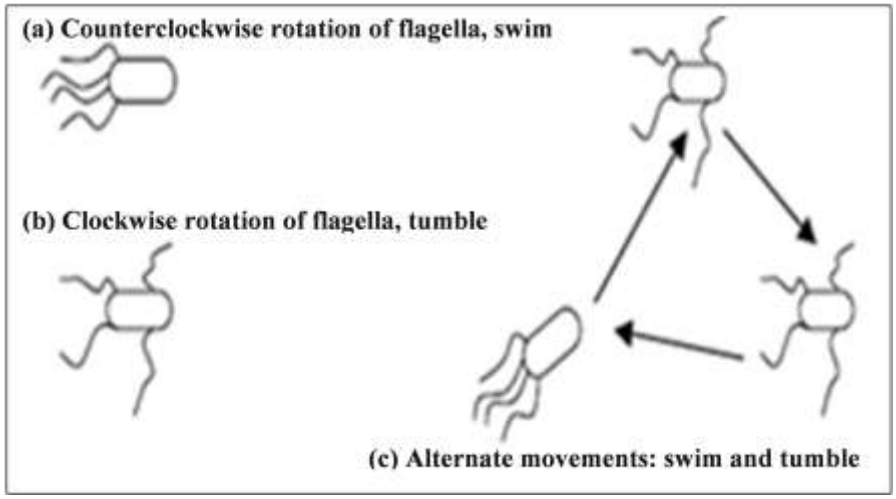


Fig. (18): Swimming, tumbling and chemotactic behavior of *E. coli.*

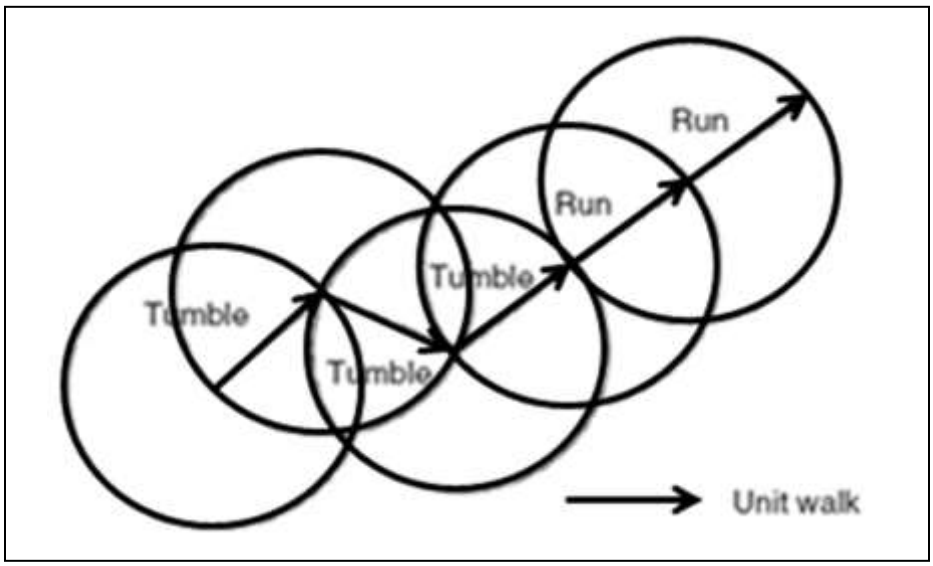


Fig. (19): Tumble and Run.

After one step move, the position of the *i* th bacterium can be represented as:

$\Theta_i(j+1, r, \ell) = \Theta_i(j, r, \ell) + C(i)\,\emptyset(j)$ **(1)**

Where, $\Theta_i(j, r, \ell)$ indicates the position of the *i* th bacterium at the *j* th chemotactic step in the *r* th reproductive loop of the ℓ th elimination and dispersion event; *C(i)* is the length of a unit walk, which is set to be a constant; $\emptyset(j)$ is the direction angle of the *j* th step. When its activity is Run, $\emptyset(j)$ is the same with $\emptyset(j-1)$; Otherwise, $\emptyset(j)$ is a random angle generated within a range of $[0, 2\pi]$. With the activity of Run or Tumble taken at each step of the chemotactic process, a step fitness, denoted *as* $J_i(j, r, \ell)$ will be evaluated.

3.2 Reproduction:

The total fitness of each bacterium is calculated as the sum of the step fitness during its life, i.e $\sum_{j=1}^{Nc} J_i(j, r, \ell)$ which is obtained after all chemotactic steps, where Nc is the maximum step in a chemotactic process. All bacteria are sorted in reverse order according to their fitness.

The least healthy bacteria die, but each of the healthier bacteria (i.e., those yielding the lower value of the objective function) asexually split into two bacteria, which are placed in the same location. This makes the population of bacteria constant in each chemotactic process [10, 12, 45, 49].

3.3 Elimination-dispersal:

The dispersion event happens after a certain number of reproduction processes. In the local environment, the life of a population of bacteria changes either gradually by consumption of nutrients by consumption of nutrients or suddenly due to some other influence. Events can occur such that all the bacteria in a region are killed or a group is dispersed into a new part of the environment. To simulate this phenomenon in bacterial foraging optimization (BFO), some bacteria are liquidated at random *with a very small probability (*ped*)* while the new replacements are randomly initialized over the search space [2, 45, 49].

3.4 Guidelines for bacterial foraging algorithm parameter choices:

These guidelines for BFO algorithm parameter as mentioned in [3, 21, 32, 33] are:

1) Size of population 'S**':** Increasing the size of S can significantly increase the computational complexity of the algorithm. But, it is more likely at least some bacteria near an optimum point should be started and over time. It is then more likely that many bacteria will be in that region, due to either chemotaxis or reproduction.

2) Length of chemotactic step 'C(i)**':** The step size of each generation in BFO is the main determining factor for accuracy as well as convergence of global best optima. BFO with fixed step size suffers from the following problems: If the C(i) values are too large, then it may simply miss possible local minima by swimming through them without stopping. On the other hand, if the C(i) values are too small, convergence can be slow, but if the search finds a local minimum it will typically not deviate too far from it. C(i) can be treated as a type of "step size" for the optimization algorithm.

3) Chemotactic step ' *Nc* **':** If the size of *Nc* is chosen to be too short, the algorithm will generally rely more on luck and reproduction, and in some cases, it could more easily get trapped in a local minimum (premature convergence).

4) Limit the length of a swim *or* maximum number of steps, ' *Ns* **':** It creates a bias in the random walk (which would not occur if Ns = 0), with large values tending to bias the walk more in the direction of climbing down the hill.

5) Reproduction number ' *Nre* **':** If *Nre* is too small, the algorithm may converge prematurely. Larger values of *Nre* clearly increase computational complexity.

6) Elimination and dispersal number ' *Ned* **':** A low value for Ned dictates that the algorithm will not rely on random elimination-dispersal events to try to find favorable regions. A high value increases computational complexity but allows the bacteria to look in more regions to find good nutrient concentrations. If it is chosen appropriately, it can help the algorithm jump out of local optima and into a global optimum.

Fig. (20) shows the flowchart of Bacteria Foraging Optimization Algorithm.

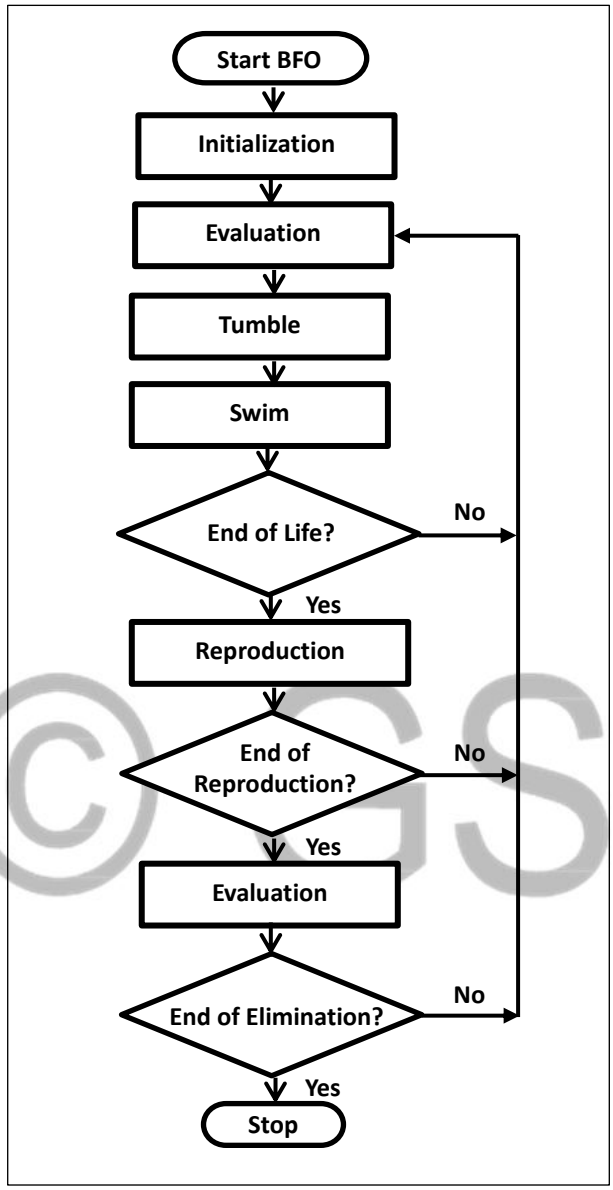


Fig. (20): Flowchart of bacteria foraging optimization algorithm.

3.5 Advantages of BFO algorithm:

- It does not require penalty function for handling constrained problems.
- It searches a population of points in parallel, not just a single point.
- BFO is designed to tackle non gradient optimization problems and to handle complex and non-differentiable objective functions [4].
- BFO has been successfully applied on various applications like option model calibration, image processing, RFID network scheduling, and many other applications.

Although these advantages of BFO algorithm, very few researchers have tried to improve the quality of optimal solution over multimodal and high dimensional functions using BFO. Researchers have tried to improve quality of solution by fusion of BFO with particle swarm optimization (PSO) and genetic algorithm (GA). Experimental results show that BFO gives poor performance for multimodal and high dimensional functions as compared to other optimization techniques like GA and PSO [32].

4. Harmony Search

This section presents a brief overview of the HS. Harmony Search (HS) was first developed by Zong Woo Geem et al. in 2001, though it is a relatively new metaheuristic algorithm, its effectiveness and advantages have been demonstrated in various applications. This section presents the harmony search (HS) algorithm for solving optimization problems. It is inspired by the natural musical performance process that occurs when a musician searches for a better state of harmony. When a musician is improvising, he or she has three possible choices [22, 42]:

(1) Playing any one famous pitch of music from his (or her) memory, (2) Playing an adjacent pitch of one pitch from his (or her) memory, and/or (3) Compose new or random notes.

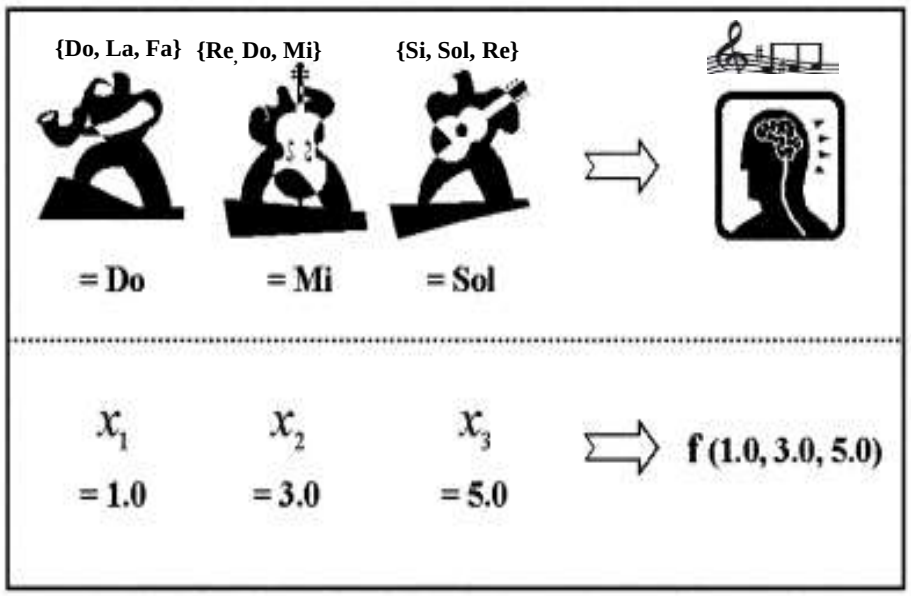


Fig. (21): Analogy between music improvisation and problem optimization.

Similarly, when each decision variable chooses one value in the HS algorithm, it follows any one of three rules:

(1) Choosing any one value from HS memory (defined as memory considerations),
(2) Choosing an adjacent value of one value from the HS memory (defined as pitch adjustments), and
(3) Choosing totally random value from the possible value range (defined as randomization).

Just like musical harmony is improved time after time, solution vector is improved iteration by iteration. Although HS searches the optimal solution by considering multiple solution vectors as in GA, its reproduction process is different from GA. While GA generates a new offspring from two parents in the population, HS generates it from all of the existing vectors stored in HM.

4.1 The steps of harmony search:

In general, HS is performed based on the following five steps as described in [31, 39, 41, 42].

Step1. Initialize the problem and algorithm parameters.
Step2. Initialize the harmony memory (HM).
Step3. Improvise a new harmony from the HM.
Step4. Update the HM.
Step5. Repeat Steps 3 and 4 until the termination criterion is satisfied.
More explanation of these steps is in the following manner.

Step1. Initialize the problem and algorithm parameters:

These parameters are:

1. Harmony: It is the same as gene in the genetic algorithm. It is the set of the values of all the variables of the objective function [7].
2. The harmony memory size (HMS), or the number of solution vectors in the harmony memory;
3. Harmony memory considering rate (HMCR);
4. Pitch adjusting rate (PAR); and
5. The number of improvisations (NI), or stopping criterion.

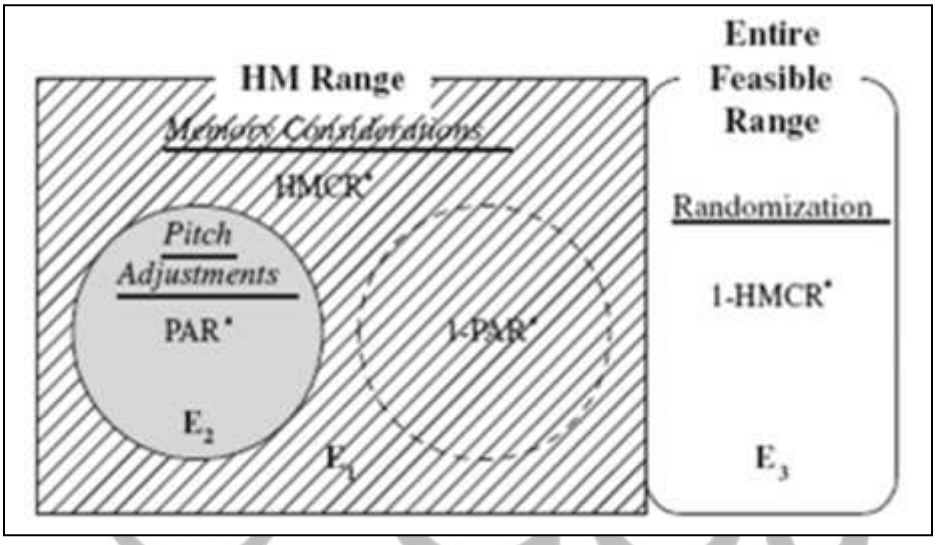


Fig. (22): New harmony improvisation concept.

Step2. Initialize the harmony memory (HM):

Harmony search as mentioned previously is mimicking the improvisation process of musicians with an intelligent way as seen in the following figure.

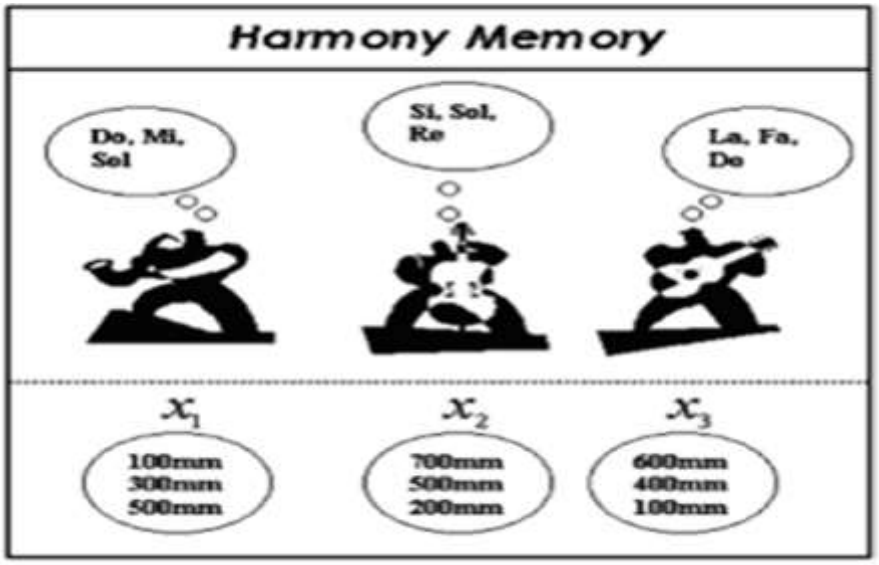


Fig. (23): Structure of harmony memory.

The harmony memory (HM) is a matrix of solutions with a size of HMS; where each harmony memory vector represents one solution as can be seen in **Fig. (24)**. In this step, the solutions are randomly constructed and rearranged in a reversed order to HM, based on their objective function values. The initialized HM can be described as follows:

$$\begin{bmatrix} x_1^{\,1} & x_2^{\,1} & \dots & x_N^{\,1} & | & f(X^1) \\ x_1^{\,2} & x_2^{\,2} & \dots & x_N^{\,2} & | & f(X^2) \\ \vdots & \vdots & \ddots & \vdots & | & \vdots \\ x_1^{\,HMS} & x_2^{\,HMS} & \dots & x_N^{\,HMS} & | & f(X^{HMS}) \end{bmatrix}$$

Fig. (24): The HM matrix is filled with as many randomly generated solution vectors as the HMS.

Step3. Improvise a new harmony from the HM:

In the memory consideration, the value of the first decision variable (x_1') for the new vector is chosen from any one of the values in the specified HM's range ($x_1'^1$ - $x_1'^{HMS}$). Values of the other decision variables (x_2' , x_3', … , x_N') are chosen in the same manner. The HMCR, which varies between 0 and 1, is the rate of choosing one value from the historical values stored in the HM, while (1- HMCR) is the rate of randomly selecting one value from the possible range of values.

$$x_1' \longleftarrow \begin{cases} x_i' \in \{x_i^1, x_i^2, \dots, x_i^{HMS}\} & \text{with probability HMCR,} \\ x_i' \in X_i & \text{with probability (1- HMCR).} \end{cases} \quad \textbf{(2)}$$

Every component obtained by the memory consideration is examined to determine whether it should be pitch-adjusted.

$$\text{Pitch adjusting decision for } x_1' \longleftarrow \begin{cases} \text{Yes} & \text{with probability PAR} \\ \text{No} & \text{with probability (1- PAR).} \end{cases} \quad \textbf{(3)}$$

The value of (1- PAR) sets the rate of doing nothing. If the pitch adjustment decision for x_i' is YES, x_i' is replaced as:

$$x_i' \longleftarrow x_i' \pm \text{r. bw}, \quad \textbf{(4)}$$

Where:

bw is an arbitrary distance bandwidth, and

r is a random number between 0 and 1.

If HMCR is too low, only few elite harmonies will be selected. As a result, HS will converge slowly. Of course an HMCR value of 1.0 is not recommended because the exploration of the entire feasible range will be obstructed and optimization will fail. Typical values of HMCR are always greater than 70%.

Pitch adjustment is similar to mutation procedure in GA. Although PAR usually takes small values (≈5%), recent literature regards PAR as a very important factor, responsible for the convergence. Moreover, recent studies suggest dynamic change of PAR and bw during the performance of the Algorithm. Pitch Adjusting is the local search mechanism which controls the ability for fine-tuning. Because of the importance of PAR, some scientists think of increasing its percentage even up to 50%.

Step4. Update the HM:

If the new harmony vector, $x_i'= (x_1' , x_2', \dots , x_N')$ is better than the worst harmony in the HM, judged in terms of objective function value:

1. The new harmony is included in the HM (similar with parents in GA for the creation of new, even better harmonies) and,
2. The existing worst harmony is excluded from the HM [7, 34, 39].

Step5. Repeat Steps 3 and 4 until the termination criterion is satisfied:

1. If the stopping criterion (maximum number of improvisations) is satisfied, computation is terminated.

2. Otherwise, **Steps 3** and **4** are repeated.

The optimization steps of HS are sketched basically as shown in **Fig. (25)**.

Both memory considering and pitch adjusting ensure that the good local solutions are retained while the randomization and harmony memory considering will explore the global search space effectively; i.e., the HMCR and PAR parameters help the algorithm to find globally and locally improved solutions, respectively [3, 22, 39].

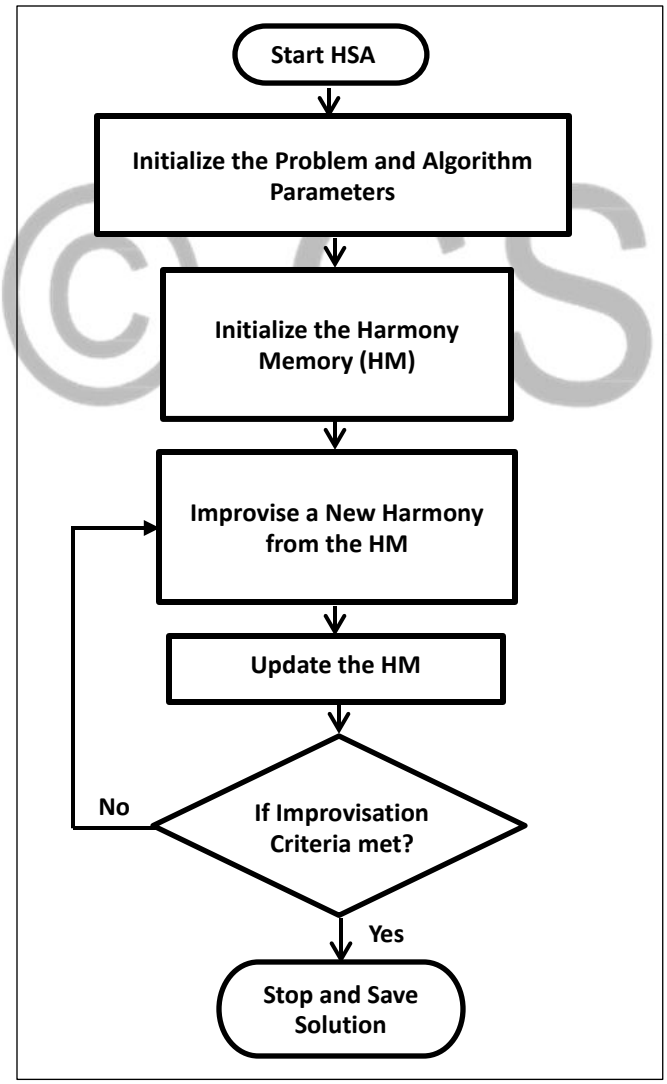


Fig. (25): Basic flowchart diagram for harmony search algorithm.

4.2 Advantages of harmony search:

This algorithm presents several advantages with respect to traditional optimization techniques such as the following:

a) HS algorithm imposes fewer mathematical requirements and it does not require penalty function for handling constrained problems. So, it is conceptually simple and relatively easy to implement.

b) As the HS algorithm uses stochastic random searches, derivative information is also unnecessary.

c) The HS algorithm generates a new vector, after considering all of the existing vectors.

d) HS algorithm is highly effective in different situations; where many initial vectors interact to produce a large number of possible solutions.

However, the effectiveness of the algorithms is analyzed in terms of computational time and solution quality; these features increase the flexibility of the HS algorithm and produce better solutions [31, 35, 39, 42].

5. A comparison of GA, BFO, and HS

There are algorithmic analogies of genetic, bacterial, and harmony for multi-objective optimization problems.

(1) The initial population and parameters:

GA generates initial population range for all design variables with random values at once. But, BFA and HS generate initial design variable with random values for each variable. Basically, the selection process of parameter's values for GA is more difficult than BFO and HS.

(2) The fitness of population:

In this phase, suitability of each solution from the initial set is determined by the problem's function. This function is called fitness function. There are analogies between the fitness functions for GA, HS and the nutrient concentration function for BFO. Each of them is a type of "landscape". The fitness functions (for GA and HS) and the nutrient concentration function are not the same (the first represents likelihood of survival for given phenotypic characteristics, the second, represents best harmony, whereas the last represents nutrient/ noxious substance concentrations or for other foraging predator/ prey characteristics).

GA uses the penalty function for handling constrained problem. Also, HS may be used this function to transform the constrained optimization problem into unconstrained problem. But, BFO does not need to it.

In this study, the weights of objectives are equals; for all multi-objective optimization problems; to transform into single objective optimization problem.

(3) Operation:

For *genetic operation*, the goal is the generation of new population from the existing population with the examination of fitness values of chromosomes and application of genetic operators. The genetic operators provide the basic search mechanism of the GA, which are: selection, crossover, and mutation. These operators are called reproduction of genetic algorithm.

But in *bacterial operation*, the bacterial behavior of E. coli can be explained by three processes namely, chemotaxis, reproduction, and elimination and dispersal.

The *harmony operation* is basically depended on the harmony memory and its parameters. The reproduction of harmony algorithm consists of two operators: improvise a new harmony from the HM, and updating the HM.

It is clear that the selection of children in genetic, reproduction of bacteria, and reproduction of harmonics gain a selective advantage for reproduction in the most favorable environments. Crossover and bacterial splitting (the children are the same concentration, whereas with crossover they generally end up in a region around their parents on the fitness landscape). Crossover represents mating and resulting differences in offspring. Something cause in the bacterial algorithm, the least healthy bacteria die and the other healthiest bacteria split in two bacteria; which are placed at the same location. Moreover, mutation represents mutation gene and the resulting phenotypical changes, not physical dispersal of an

environment. Pitch adjustment rate in HS is similar to mutation probability in GA. But, mutation in GA, and elimination and dispersal in BFA are not equivalent. Where, each algorithm has its own distinguishing features.

(4) Stopping criteria:

If the termination condition is satisfied, the algorithm ends, otherwise it returns to the suitable step.

(5) The resulted solution:

Through observing the processes of GA, BFA, and HS, it is noticed for a single function optimization problem that: Genetic and bacterial emphasize the local search ability, but the harmony performs high searching ability on global search.

In addition, the solution set in BFA is a single bacterium, and it basically finds optimal solution on its own. On the contrary, the solution set in GA is a chromosome, which carries the solution as gene, but it searches the optimal solutions by exchanging the information with others. HS depends on a matrix of solutions with a size of harmony memory, where each harmony memory vector represents one solution. The role of harmony in HS algorithm is the same as gene in the genetic algorithm [21, 33].

6. Penalty function method

Penalty function method is used for handling constrained problems in this research.

There are two types of methods:

- Interior penalty methods, also known as barrier methods
- Exterior penalty methods

We will reformulate **F(x)** such that the constraints are part of the objective function. We also want to insure that the two essential conditions of Lagrange's method hold:

1. If an optimal solution exists that requires the constraint to be tight, then any perturbation away from constraint activity should be heavily penalized.
2. If an optimal solution dictates the constraint be loose at optimality, the problem should behave as though it were unconstrained by that constraint.

The reformulated objective function:

Minimize: $\mathbf{P}(\bar{\mathbf{x}}, \mathbf{s}, \mathbf{r}) =$

$$\mathbf{F(x)} + \mathbf{s} \sum_{k=1}^{p} (r_k \, \phi(g_k(\bar{\mathbf{x}})) + \left(\sum_{i=1}^{m} t_i \, \Psi(h_i(\bar{\mathbf{x}})) \right) \qquad (5)$$

Where:

F(x) is the original objective function,

s = Penalty parameter,

$\mathbf{t_i}$ = Scale factor for equality constraints,

$\mathbf{r_k}$ = Monotonically increasing or decreasing inequality scale factor,

$\boldsymbol{\phi}(g_k(\bar{\mathbf{x}}))$: Penalty function involving each original inequality constraint,

$\boldsymbol{\Psi}(h_i(\bar{\mathbf{x}}))$: Penalty function involving each original equality constraint.

This is a generalized equation that represents both interior and exterior methods. The nature of **s**, **r** and $\boldsymbol{\phi}$ in the general formulation depends on whether one wishes to begin with a feasible solution and iterate toward optimality, or start with an infeasible solution and proceed toward both optimality and feasibility.

The equality scale factor, $\mathbf{t_i}$, is the penalty for violation of an equality constraint. In the infinite barrier method, $\mathbf{t_i}$ would be set to infinity, approximated numerically by some large number such as 10^{20}. This method does not work well with inequality constraints.

These are some important notes:

1) Exterior penalty methods start at optimal but infeasible points and iterate to feasibility as r ⟶ inf.

2) Interior penalty methods start at feasible but sub-optimal points and iterate to optimality as r ⟶ 0.

3) The methods are iterative Sequential Unconstrained Minimization Techniques.

4) We start with r reasonable because it makes the problem better conditioned (less sensitive to numerical round off, etc.) and hence easier to solve.

5) Exterior methods are generally more robust [20, 23, 40].

7. Conclusion

This paper presented an overview of the genetic algorithm, bacterial foraging optimization, and harmony search with a goal of providing useful fundamental concepts, analysis, and a comparison of them. The main goal of HS algorithm is to avoid the disadvantages of iterative improvement and a local optima problem for GA and BA.

Acknowledgment

Thanks be to *ALLAH* for this guidance and support in showing me the path.

References

[1] Alia Youssef Gebreel, "An adaptive interactive multi-objective optimization approach based on decision neural network", *International Journal Scientific & Engineering Research (ISSN 2229- 5518)*, Vol. 7, No. 8, PP. 1178- 1185, (2016).

[2] Amrinder Singh, and Sonika Jindal" A systematic way for image segmentation based on bacteria foraging optimization technique (Its implementation and analysis for image segmentation)", *International Journal of Computer Science and Information Technologies*, Vol. 5, No.1, PP.130-133, (2014).

[3] A. Zeblah, E. Chatelet, M. El Samrout, F. Yalaoui, and Y. Massim, " Series-parallel power system optimisation using a harmony search algorithm", *Int. J. Power and Energy Conversion,* Vol. 1, No. 1, PP. 15- 30, (2009).

[4] Bobbinpreet Kaur, and Raju Sharma, " Image segmentation using RGB decomposition and modified bacterial foraging optimization ", *International Journal of Computational Engineering Research,* Vol. 03, No. 5, PP. 50- 55, May **(**2013).

[5] Chunguo Wu, Na Zhang, Jingqing Jiang, Jinhui Yang, and Yanchun Liang, "Improved bacterial foraging algorithms and their applications to job shop scheduling problems", B. Beliczynski et al. (Eds.): ICANNGA, Part I, LNCS 4431, Springer-Verlag Berlin Heidelberg, PP. 562 – 569, (2007).

[6] Christopter R. Holck, Jeffery A. Joines, and Michael G. Kay, "A genetic algorithm for function optimization: A MATLAB implementation ", North Carolina State University, 2008.

[7] D. Sai Kumar, and D. Dedeepya, " A Meta-Heuristic approach to feature selection using harmony search ", *International Journal of Advanced Research in Computer Science*, ISSN No. 0976-5697, Volume 5, No. 7, September-October (2014).

[8] Ganesan R., Shanmuga Priyan P., and Kerana Hanirex D.," A formal machine learning or multi objective decision making system for determination of weights", *International Journal of Computer Applications Technology and Research*, Vol. 4, No. 4, PP. 287 - 290, (2015).

[9] Hong- Zhong Huang, Zhigang Tian, and Ming J. Zuo, "Intelligent interactive multiobjective optimization method and application to reliability optimization", *IIE Tranction*, 37, PP. 983- 993, (2005).

[10] H. Noormohamadi, A. A.Gharaveisi, M. Suresrafil, "A novel bacterial foraging algorithm for optimization problems", *Advances in electrical engineering systems*, Vol. 1, No. 1, March, (2012), www.Worldscience publisher.org.

[11] Internet: Brandon Madura, Markus Borneman, and Chris Kam, "Genetic algorithm", http:// www. youtube.com/watch? v= ZwYV11a _HQ.

[12] Internet: chapter 5, "Bacterial foraging optimization algorithm (BFOA)", PP. 62-73.

[13] Internet, Chapter 18, Cooperative foraging and search.

[14] Internet: Genetic algorithm, https: //www .doc.ic.ac.uk/~ nd/surprise _96 /journal/vol1/hmw/articlel.html.

[15] Internet: Genetic algorithms, Tutorials Point (I) Pvt. Ltd, contact @tutorialspoint.com, 2016.

[16] Internet: http:// hungrydevelop. blogspot. Com / 2013/ 10/ genetic-algorithm –implementation -in-matlab.html, In Soft Computing by Sagar Gopani // 1/03/2014 11:47:00 PM // Leave a Comment.

[17] Internet: IV Genetic algorithm, http.
[18] Internet: J. S. Roger Jang, "Derivative-free optimization", CS Dept., Tsing Hua Univ., Taiwan, http://www.cs.nthu.edu.tw/~ jang, jang@cs.nthu.edu.tw.
[19] Internet: "Multi-objective optimization", Wikipedia, the free encyclopedia, 2015.
[20] Internet: Penalty function method, http://web .engr. oregonstate. edu/~paasch/classes/me517/week7/penalty.html.
[21] Internet: Tai-Chen Chen, Pei-Wei Tsai, Shu-Chuan Chu, and Jeng-Shyang Pan, "A novel optimization approach: Bacterial-GA foraging" *National Kaohsiung University of Applied Sciences*, Taiwan, Cheng Shiu University, Taiwan, *pwtsai@bit.kuas.edu.tw, scchu@csu.edu.tw, jspan@cc.kuas.edu.tw*.
[22] Internet: Xin-She Yang," Harmony search as a metaheuristic algorithm", Email: xy227@cam.ac.uk.
[23] J. A. Snyman, Nielen Stander, and W. J. Roux, "A dynamic penalty function method for the solution of structural optimization problems", *Appl. Math. Modelling*, Vol. 18, PP. 453- 460, August, (1994).
[24] Jason Brownlee, "Clever algorithms: Nature-inspired programming recipes", Ph.D. at Swinburne University, 2011, http://www.Clever Algorithms.com.
[25] Javier Sanchis, Miguel A. Martinez, and Xavier Blasco, "Integrated multiobjective optimization and a priori preferences using genetic algorithms", *Information Sciences, An International Journal*, www. Elsevier.com/locate/ins, 178, PP. 931- 951, (2008).
[26] Jian-Bio Yang, Pratyush Sen, "Preference modeling by estimating local utility functions for multiobjective optimization", *European Journal of Operational Research*, 95, PP. 115- 138, (1996).
[27] John Paul T. Yusiong and Prospero C. Naval Jr, "Training neural networks using multiobjective particle swarm optimization", L. Jiao et al. (Eds.): ICNC, Part I, LNCS 4221, PP. 879–888, Springer-Verlag Berlin Heidelberg, 2006.
[28] Kaisa M. Miettinen, "Nonlinear multiobjective optimization", Kluwer Academic Publishers, 1998, and Fourth Printing 2004.
[29] Kalyanmoy Deb, "Multi-objective optimization using evolutionary algorithms", John Wiley & Sons, Ltd, 2001.
[30] 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).
[31] Kang Seok Lee, and Zong Woo Geem, "A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice", *Applied Mathematics and Computation*, www.elsevier.com/locate/amc, 188, PP. 1567–1579, (2007).
[32] K. M. Bakwad, S. S. Pattnaik, B. S. Sohi, S. Devi, B. K. Panigrahi and Sastry V. R. Gollapudi, "Multimodal function optimization using synchronous bacterial foraging optimization technique", *IETI Journal of Research*, Vol. 56, No. 2, PP. 80- 87, MAR- APR (2010).
[33] Kevin M. Passino, "Bacterial foraging optimization", *international journal of swarm intelligent research*, Vol.1, No. 1, PP. 1- 16, January-March (2010).
[34] Leandro dos Santos Coelho, and Viviana Cocco Mariani, "An improved harmony search algorithm for power economic load dispatch", *Energy Conversion and Management*, 50, PP. 2522–2526, (2009).
[35] Li Hong-qi, LI Li, Tai-hoon Kim, and XIE Shao-long," An improved PSO-based of harmony search for complicated optimization problems", *International Journal of Hybrid Information Technology*, Vol.1, No. 1, PP. 57- 64, January, (2008).
[36] Livjeet Kaur, Mohinder Pal Joshi, "Analysis of chemotaxis in bacterial foraging optimization algorithm", *International Journal of Computer Applications (0975 – 8887),* V. 46, No.4, PP. 18- 23, May (2012).
[37] Maridass Balasubramanian, Marissa A. Paglicawan, Zhen-Xiu Zhang, Sung Hyolee, Zhen-Xiang Xin, and Jin Kuk Kim," Prediction and optimization of mechanical properties of polypropylene/waste tire powder blends using a hybrid Artificial Neural Network-Genetic Algorithm (GA-ANN)", *Journal of Thermoplastic Composite Materials*, Vol. 21, PP. 51- 69, (2008), http://www.sagepublications.com.
[38] MATLAB software, Version 7.0, 2004.
[39] M. Mahdavi, M. Fesanghary, and E. Damangir, ''An improved harmony search algorithm for solving optimization problems'', *Applied Mathematics and Computation*, 188, PP. 1567- 1579, (2007).
[40] Mokhtar S. Bazzraa, Hanif D. Sherali, C. M. Shetty, "Nonlinear programming: Theory and algorithms", John Wiley & Sons, Inc., 1979, 1993, and 2006.
[41] M. Tamer Ayvaz, "Application of harmony search algorithm to the solution of groundwater management models", *Advances in Water Resources*, 32, PP. 916–924, (2009).
[42] Parikshit Yadav, Rajesh Kumar, S.K. Panda, and C.S. Chang, "An improved harmony search algorithm for optimal scheduling of the diesel generators in oil rig platforms", *Energy Conversion and Management*, 52, PP. 893–902, (2011).
[43] Petr Fiala, "Multiobjective de novo linear programming", Acta Univ. Palacki. Olomuc., Fac. rer. nat., *Mathematica*, Vol. 50, No.2, PP. 29–36, (2011).

[44] Rajib Kumar Bhattacharjya, "Introduction to genetic algorithms", *Indian Institute of Technology Guwahati*, Email: rkbc@iitg.ernet.in, November 2013.

[45] Reza-Salehi, Behrooz-Vahidi, Naeem-Farokhnia and Mehrdad-Abedi, “Harmonic elimination and optimization of stepped voltage of multilevel inverter by bacterial foraging algorithm”, *Journal of Electrical Engineering & Technology*, DOI: 10.5370/ JEET .2010. 5. 4. 545 Vol. 5, No. 4, PP. 545-551, (2010).

[46] Roberto Battiti, Fellow, IEEE, and Andrea Passerini, ''Brain–computer evolutionary multiobjective optimization: A genetic algorithm adapting to the decision maker'', IEEE *Transactions on Evolutionary Computation*, Vol. 14, No. 5, PP. 671-687, October (2010).

[47] Seema Mane, S. S. Sonawani, Sachin Sakhare, and P. V. Kulkarni, ''Multi-objective evolutionary algorithms for classification: A review'', *International Journal of Application or Innovation in Engineering & Management (IJAIEM)*, ISSN 2319 - 4847, Vol. 3, No. 10, PP. 292- 297, October (2014).

[48] Ulrich Bodenhofer, "Genetic algorithms: Theory and applications", Lecture notes, Third Edition—Winter 2003/2004.

[49] 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).