Journal of Theoretical and Applied Information Technology
15th December 2020. Vol.98. No 23

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

IMPROVED MOTH FLAME OPTIMIZATION BASED ON HARRIS HAWKS FOR GENESSELECTION

RUBA ABU KHURMA[1], IBRAHIM ALJARAH[2], AND AHMAD SHARIEH[3]

[1]Phd student, King Abdullah II School for Information Technology, The University of Jordan, Jordan

[2]Associate prof, King Abdullah II School for Information Technology, The University of Jordan, Jordan

[3]Full prof, King Abdullah II School for Information Technology, The University of Jordan, Jordan

[1]RBA9150539@fgs.ju.edu.jo, {[2]i.aljarah, [3]sharieh}@ju.edu.jo

ABSTRACT

Microarray technology is a major shift in the medical and diagnostic fields. Gene expression data are coded by a large number of genes and a limited number of patient's samples. This causes a challenging problem for the gene selection (GS) methods to specify the most relevant and reliable genes for cancer diagnosis. Recently, meta-heuristic (MH) algorithms have made an evident contribution in handling the gene expression data. A great expansion has been achieved in developing robust and efficient cancer diagnostic systems. In this chapter, a hybrid MH-MH wrapper, called IMFOHHO is proposed for the GS problem. The proposed approach is based on hybridizing the Moth Flame Optimization (MFO) algorithms and Harris Hawks Optimization (HHO) to improve the exploration and the exploitation phases. The main purpose is to integrate the excellent features of both algorithms in one model and overcome their limitations when experienced in the gene search space. Combining two swarm systems in one model can achieve strong exploration for the gene space and ensure the diversity of solutions. The performance of the proposed method has been evaluated using ten gene expression data sets. The comparative study demonstrates that the IMFOHHO model enhances the classification performance without increasing the computational complexity.

Keywords: *Genes selection* (GS), *Microarray*, *Harris Hawks Optimization (HHO), Moth Flame Optimization (MFO), Classification.*

1. INTRODUCTION

Microarray is a microchip technology to study the sequences of genes. This technology was developed at the end of the nineties and has become very popular so far [1]. There are several ways to create these slides. The first method is to use Spotting robots that install the complement chains (cDNA) in the slots (probes) in the slice. The second method is to construct complementary chains directly on the slide (in-sito Spotting). The principle of cell function, in general, is that the activity of each gene leads to the production of a protein that plays several roles in the cell. Thus, the greater the activity of a gene, the higher the amount of mRNA (messenger RNA) produced, and thus the greater the proportion of proteins manufactured by this gene [2]. The easiest way to identify the active genes in the cell is to extract the mRNA present in the cell and measure its ratio. An important characteristic of mRNA is hybridization. This means that a complement copy can be made to it (cDNA) using the base molecules C, G, T, and A. Microarray chips are used to measure the percentage of mRNA that has been attached (hybrid) with cDNA complementary chains. Furthermore, they can help biologists to measure the activity of the gene and to know the active and passive genes (cancer, for example). Thus, they can analyze the state of a cell at the molecular level and find the relations between the genes and the biological conditions.

DNA microarray technology generates gene expression data sets. The columns represent the genes and the instances represent the clinical patients' samples [3]. Gene expression data sets

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

enable the study of cells, tissues, and screen the potential tumor markers based on the molecule information. Thus, they are considered precious tools in functional genomics and biomarker discovery. However, they are characterized by high dimensionality (a large number of genes) with a small number of instances. Ordinarily, high dimensionality causes the abundance of noise, spurious information, irrelevant and redundant genes. From a molecule biologist's point of view, this poses a challenge because it increases the cost of the clinical process as more specialized resources are required. Moreover, it complicates the identification of diseases because many irrelevant genes may be used in the generated gene expression. From a computational intelligence specialist's point of view, increasing the dimensionality (the curse of dimensionality) may mislead the learning algorithm and adversely affects the generalization process as many noisy genes may be used in building the learning model.

Gene selection (GS) is a critical preprocessing step for accurate classification of diseases. The main purpose of GS is to simplify the quality of large-scale databases by retaining the salient features (most discriminative genes). These genes are a key marker for the recognition and classification of specific diseases [4]. Gs eliminates the uninformative genes without sacrificing the learning performance. A reliable GS generates a compact gene expression data that simultaneously increases the pattern recognition performance. On the other side, dealing with all genes is incomprehensible and time-consuming. GS helps the biologists to easily interpret the yielded gene patterns and discover the appropriate therapy for these genes [5].

GS methods are broadly classified into two groups: filters and wrappers. Filter methods use the interior characteristics of the data. They decide the redundancy and relevancy of the genes based on the associations between the genes and between the genes and the concept class. Widely used examples of filter methods are ReliefF, Chi-square, and Information gain [6]. Wrapper methods compromise two main processes: the search and the evaluation. The search is triggered by a specified search algorithm. It generates the candidate gene subsets that make up the gene solutions space. In the evaluation process, a learning algorithm is used to evaluate the goodness of a gene subset. This means that each gene subset is used to build a model through an internal training process. For this reason, wrappers often produce more accurate performance results compared to filters but they need more computational time [39].

There are different methods to search within the gene solutions search space. Brute force methods, for example, create the complete gene subset space. They traverse all the generated gene subsets that are potential solutions for GS problem. This costs an expensive search process and the time will grow exponentially as the number of genes increases. The exhaustive search makes the GS an Np- hard problem. MH algorithms are randomized methods that iteratively evaluate and update each solution until a stopping condition is satisfied. The MH algorithm may miss the exact optimal solution but they can return a promising near-optimal solution within a reasonable running time.

Swarm Intelligence (SI) is a category of MH algorithms that is inspired by the social behavior of creatures [38]. Originally, SI's simulate the way of exchanging information between the swarm's members to survive and capture prey. This collective behavior is turned into a mathematical model in which the members of a swarm are the candidate solutions (population) of the optimization problem. The population is reconstructed at each iteration in such a way the fittest solutions lead the swarm toward optimality. All SI algorithms, whatever their mathematical models and inspirations have two primary phases: diversification (exploration) that indicates the global search and intensification (exploitation) which refers to the local search. The main advantages of SI methods: scalability, self-organization, adaptability, flexibility, robustness, and simplicity. However, any SI method has some shortcomings that affect its performance. SIs still suffer from local optima problem and the high computational cost. Several modification strategies have been adopted to overcome the SI's drawbacks. In the literature, the modification strategies have been classified into new operators, update strategies, new fitness functions, new initialization, multi-objective, parallelism, and hybridization. In specific, many modification strategies have been effectively used to improve the performance of SI's for solving GS problems. They have achieved superior results in disease classification. In [7] and [8], the reset update strategy was used to improve the Binary Particle Swarm optimization algorithm (BPSO) by allowing the stagnated global best to

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

jump from local minima. A new Boolean algebra operation was used to update the BPSO [9]. A hybrid filter-wrapper model compromised of a combination of Information Gain (IG), Correlation-based Feature Selection (CFS), and IBPSO was introduced in [10]. The CuPSO was proposed in [11] which selects randomly partial solutions to be updated based on active genes in the current particle, local best, and global best. A combination of CFS filter and the Taguchi Chaotic BPSO (TCBPSO) was pro- posed in [12]. A new rule was proposed for updating the position of a particle [13]. This rule gives a lower probability for selecting a gene and a greater probability for the gene not being selected. The hamming distance was used to update the BPSO in [14]. A binary quantum operator was adopted with PSO (BQPSO) to sample around the personal best then use the average of the sampled points to update the current solution [15]. A hybrid system called GWO-ALO based on a combination of Gray Wolf Optimizer (GWO) and Ant Lion Optimization (ALO) was proposed in [16]. The main objective was to exploit the global search ability of the GWO and the local search performance of the ALO simultaneously in one system.

MFO is a recent SI algorithm which was developed by Mirjalili in 2015 [25]. The MFO simulates the natural movements known as transparent orientation. This movement allows the moths to travel long distances in a straight line by maintaining a fixed angle with the moonlight. However, moths are tricked by artificial lights that force them to move in a spiral way. In optimization, this is interrupted by exploring the search space at the early stages and exploiting around the best solution in the late stages. The MFO has proved its efficiency in solving the FS problem in different applications including medicine [26, 27,37]. However, both MFO and HHO are not applied in the genome field and they have never used to solve GS or handle the microarray data sets. HHO is a new MH algorithm which was proposed in 2019 by Heidari[28]. The HHO simulates the hunting method of predatory birds, Harris' hawks in nature. It incorporates two phases of exploration and four phases of exploitation. In the exploration stage, Hawks will randomly perch at some locations to monitor and track the movement of the rabbit. While in the exploitation stage, Hawks will make a sudden jump or a rapid diving team to exploit the intended prey area. HHO was investigated using unconstrained, unimodal, multi-modal, and composition problems. The obtained results were promising in comparison with other well- regarded optimizers. The HHO has been widely used in various applications such TD-LTE system in high-speed railway scenario [29], heterogeneous wireless networks [30], Satellite image de-noising [31] and feature selection [32]. The main research question, here, does the hybridization of the two SI algorithms, MFO and HHO improve the performance of the overall search process within a gene selection framework?

This chapter presents a novel wrapper approach for cancer disease classification in the context of microarray data sets by combining the HHO and MFO. Hybridization is a remarkable algorithmic solution to create a new SI system that integrates the merits of the integrated algorithms and nullifies each other's drawbacks. This enhances the optimization process in different ways: achieving a better trade-off between the exploration/exploitation, enhancing the convergence trends of the original algorithms, and alleviating the stagnation in local minima. A prime motivation for this chapter is to obtain the optimal gene subset for cancer classification based on hybridizing HHO and MFO algorithms. The proposed method does not only extract the excellent features from the high performance of the two algorithms. It also overcomes the limitations of HHO and the MFO algorithm to some extent. The combined two swarms ensure strong global optimization and the diversity of solutions compromised of the Cuckoo search algorithm (CS), Mutual Information (MI), entropy filters, and artificial neural network (ANN) was proposed in [17]. A combination of Grasshopper Optimization Algorithm (GOA) and Support Vector Machine (SVM) using two different system structures [18]. AntColony Optimization (ACO) [19], [20]. Several hybrid models based on Genetic Algorithm (GA) were proposed including GA-ANN-MI [21], GA-SVM [22], GA-MI-SVM [23] and GA-IG [24].

According to No-Free-Lunch theorem (NFL), there is no optimization algorithm that is considered the perfect solution for all optimization problems. Thus, the doors are still opened for the researchers to introduce new algorithmic solutions or update exiting ones by suggesting new modification technique. For example, new operators, new update strategies

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

and hybridization of different optimization algorithms. The rest of this paper is organized as follows: related background is described in Section 2. The proposed method is illustrated in Section 3. The experimental setup and results are analyzed in Section 4. Finally, Section 5 concludes the paper and recommends possible future work.

2 RESEARCH BACKGROUND

2.1 Moth Flame Optimization

Moth Flame Optimization algorithm (MFO) is a recent SI algorithm developed by Mirjalili in 2015 [25]. The MFO methodology was inspired by a transverse orientation mechanism that forces moths to follow a spiral path when they move close to a near light source. Figure 1 illustrates the behavior of moths in nature.

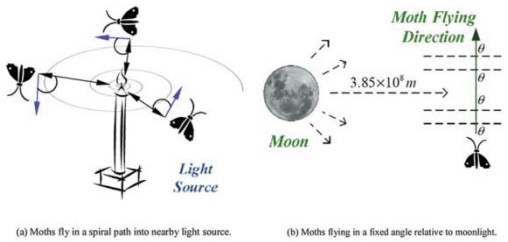

Figure. 1: The conceptual model for the movement behavior of moths

The spiral movement of moths around a flame is modeled using Eq.1 where M_i represents the *ith* moth, F_j represents the *jth* flame and S is the spiral function. A logarithmic spiral function is used to formulate the spiral movement as shown in Eq.2where D_i is the distance between the *ith* moth and the *jth* flame as shown in Eq.3, b is a constant value for determining the shape of the logarithmic spiral and t is a random number in the range [-1, 1]. The parameter $t = l$ indicates the closest position of a moth to a flame where $t = 1$ indicates the farthest position between a moth and a flame. To achieve more exploitation in the search space the t parameter is considered in the range [r,1] where r is linearly decreased throughout iterations from -1 to -2

$$M_i = S(M_i, F_j) \quad (1)$$

$$S(M_i, F_j) = D_i \times ebt \times \cos(2\pi) + F_J \quad (2)$$

$$D_i = |M_i - F_j| \quad (3)$$

In MFO, the number of flames is changed adaptively across iterations. Eq.4 shows the adaptive mechanism of MFO where l is the current number of iteration, N is the maximum number of flames and T is the maximum number of iterations. The gradual decrease in the number of flames maintains the balance between exploration and exploitation phases during the optimization process [37].

$$\text{FlameNo} = \text{round}(\text{N} - \text{l} \times (\text{N} - 1)/\text{T}) \quad (4)$$

The overall steps are presented in the pseudo-code of the basic MFO as shown in Algorithm 1

Algorithm 1 Pseudo-code of the MFO

Input: *Max iteration*, *n* (number of moths), *d* (number of dimensions).
Output: Approximated global solution.
Initialize the position of moths
While *l*<= *Max iteration* **do**
Update flame no using Eq.4
OM = FitnessFunction(M);
if *l* == 1 **then**
F = *sort*(*M*);
OF = *sort*(*OM*);
else
F = *sort*(M_{l-1}, M_l);
OF = *sort*(OM_{l-1}, OM_l);
for i = 1: n **do**
for j = 1: d **do**
Update r and t;
Calculate D using Eq.3 with Respect to the corresponding moth.
Update M (i, j) using Eqs.1 and Eqs.2 with respect to the corresponding moth.
l=l+1;

2.2 Harris Hawks Optimization

Harris Hawks Optimization (HHO) is a new SI algorithm developed by Heidari in 2019. HHO mimics the hunting strategy of predatory birds,

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Harris' hawks in nature. It has two phases of exploration and four phases of exploitation. Figure 2 shows all phases of HHO.

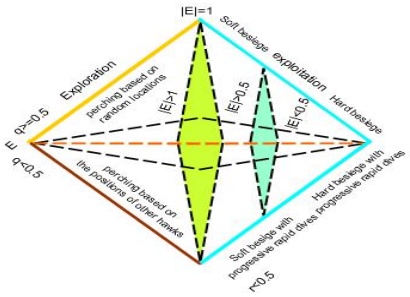

Figure. 2: Different phases of HHO

2.2.1 Exploration phase

In HHO, the Harris' hawks perch randomly on some locations and wait to detect a prey based on two strategies. By considering equal chance q for each perching strategy, they perch based on the positions of other family members (to be close enough to them when attacking) and the rabbit, which is modeled in Eq. (5) for the condition of $q < 0.5$, or perch on random tall trees (random locations inside the group's home range), which is modeled in Eq. (5) for condition of $q \geq 0.5$.

$$X(t+1) = \begin{cases} X(t) - r_1|X_{rand}(t) - 2r_2X(t)| & q \geq 0.5 \\ \left(X_{rabbit}(t) - X_m(t)\right) - r_3 \times \left(LB + r4(UB - LB)\right) & q < .5 \end{cases} \quad (5)$$

where $X(t+1)$ is the position vector of hawks in the next iteration t, $X_{rabbit}(t)$ is the position of rabbit, $X(t)$ is the current position vector of hawks, r_1, r_2, r_3, r_4, and q are random numbers inside (0,1), which are updated in each iteration, LB and UB show the upper and lower bounds of variables, $X_{rand}(t)$ is a randomly selected hawk from the current population, and X_m is the average position of the current population of hawks. The average position of hawks is attained using

$$X_m(t) = \frac{1}{N}\sum_{i=1}^{N} X_i(t) \quad (6)$$

Where $X_i(t)$ indicates the location of each hawk in iteration t and N denotes the total number of hawks. It is possible to obtain the average location in different ways, but we utilized the simplest rule.

2.2.2 Transition from exploration to exploitation

The HHO algorithm can transfer from exploration to exploitation then change between different exploitative behaviors based on the escaping energy of the prey. The escaping behavior of prey decreases the energy of prey considerably. The energy of prey is modeled as:

$$E = 2E_0\left(1 - \frac{t}{T}\right) \quad (7)$$

Where E indicates the escaping energy of the prey, T is the maximum number of iterations, and E0 is the initial state of its energy. In HHO, E0 randomly changes inside the interval (-1, 1) at each iteration.

2.2.3 Exploitation phase

According to the escaping behaviors of the prey and chasing strategies of the Harris' hawks, four possible strategies are proposed in the HHO to model the attacking stage. The preys always try to escape from threatening situations. Suppose that r is the chance of prey in successfully escaping (r <0.5) or not successfully escaping (r >=0.5) before surprise pounce. Whatever the prey does, the hawks will perform a hard or soft besiege to catch the prey. In this regard, when |E| >=0.5, the soft besiege happens, and when |E| <0.5, the hard besiege occurs.

- Soft besiege: when r >=0,5 and |E| >=0,5. This behavior is modeled by the following rules:

$$X(t+1) = \Delta X(t) - E|JX_{rabbit}(t) - X(t)| \quad (8)$$

$$\Delta X(t) = X_{rabbit}(t) - X(t) \quad (9)$$

Where $X(t)$ is the difference between the position vector of the rabbit and the current location in iteration t, r_5 is a random number inside (0,1), and $J = 2(1- r_5)$ represents the

random jump strength of the rabbit throughout the escaping procedure. The J value changes randomly in each iteration to simulate the nature of rabbit motions.

- Hard besiege: when r >=0.5 and |E| <0.5. In this situation, the current positions are updated using Eq. (10):

$$X(t+1) = X_{rabbit}(t) - E|\Delta X(t)| \quad (10)$$

- Soft besiege with progressive rapid dives: when still |E|>= 0.5 but r <0.5. This procedure is more intelligent than the previous case. To mathematically model the escaping patterns of the prey, the levy flight (LF) concept is utilized in the HHO algorithm. To perform a soft besiege, the hawks can evaluate (decide) their next move based on the following rule in Eq. (11):

$$Y = X_{rabbit}(t) - E|JX_{rabbit}(t) - X(t)| \quad (11)$$

Then, they compare the possible result of such a movement to the previous dive to detect that will it be a good dive or not. If it was not reasonable, they also start to perform irregular, abrupt, and rapid dives when approaching the rabbit based on the LF-based patterns using the following rule:

$$Z = Y + S \times LF(D) \quad (12)$$

Where D is the dimension of problem and S is a random vector by size $1 \times D$ and LF is the levy flight function, which is calculated [34].
The final strategy for updating the positions of hawks in the soft besiege phase can be performed by Eq. (13):

$$X(t+1) = \begin{cases} Y \text{ if } F(Y) < F(X(t)) \\ Z \text{ if } F(Z) < F(X(t)) \end{cases} \quad (13)$$

Where Y and Z are obtained using Eqs.(11) and (12).

- Hard besiege with progressive rapid dives
When |E| <0.5 and r <0.5, the following rule is performed in hard besiege condition:

Where Y and Z are obtained using new rules in Eqs.(15) and (16).

$$X(t+1) = \begin{cases} Y \text{ if } F(Y) < F(X(t)) \\ Z \text{ if } F(Z) < F(X(t)) \end{cases} \quad (14)$$

$$Y = X_{rabbit}(t) - E|JX_{rabbit}(t) - X_m(t)| \quad (15)$$

$$Z = Y + S \times LF(D) \quad (16)$$

where $X_m(t)$ is obtained using Eq. (6).

The pseudo code of the proposed HHO algorithm is reported in Algorithm 2.

Algorithm 1 Pseudo-code of the HHO

Input: The population size N and maximum number of iterations T
Output: The location of rabbit and its fitness value.
Initialize the random population X_i(i = 1,2…N)
while (stopping condition is not met) **do**
Calculate the fitness values of hawks
Set X_{rabbit} as the location of rabbit.
for (each hawk (X_i)) **do**
Update the initial energy E_0 and jump strength J.
Update the E using Eq. (7)
if (|E|>=1) then
Update the location vector using Eq. (5)
if (|E| < 1) **then**
if (r >=0.5 and |E| >=0:5) **then**
Update the location vector using Eq. (8)
elseif (r >=0.5 and |E| < 0:5) **then**
Update the location vector using Eq. (10)
elseif (r <0.5 and |E|>=0:5) **then** progressive rapid dives
Update the location vector using Eq. (14)
elseif (r <0.5 and |E| < 0:5) **then** progressive rapid dives
Update the location vector using Eq. (15)
Return X_{rabbit}

3 THE PROPOSED IMFOHHO FRAMEWORK

3.1 The hybrid MFOHHO Algorithm

As mentioned earlier, in this chapter we intend to hybridize the two swarm-based systems MFO

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

and HHO in one model called MFOHHO. The main objective is to complement the features of both algorithms for tackling the GS problem and enhance the classification performance of diseases. The update strategy of the HHO is merged into the structure of MFO to improve the updating of the positions of the moths in the gene space. The goals of this merging are adding more flexibility to the MFO, enhancing its capability in exploring the gene space, increasing the diversity of the positions of moths, and reaching the global optimal solution quickly. The steps of the proposed algorithm are shown inFig.3.

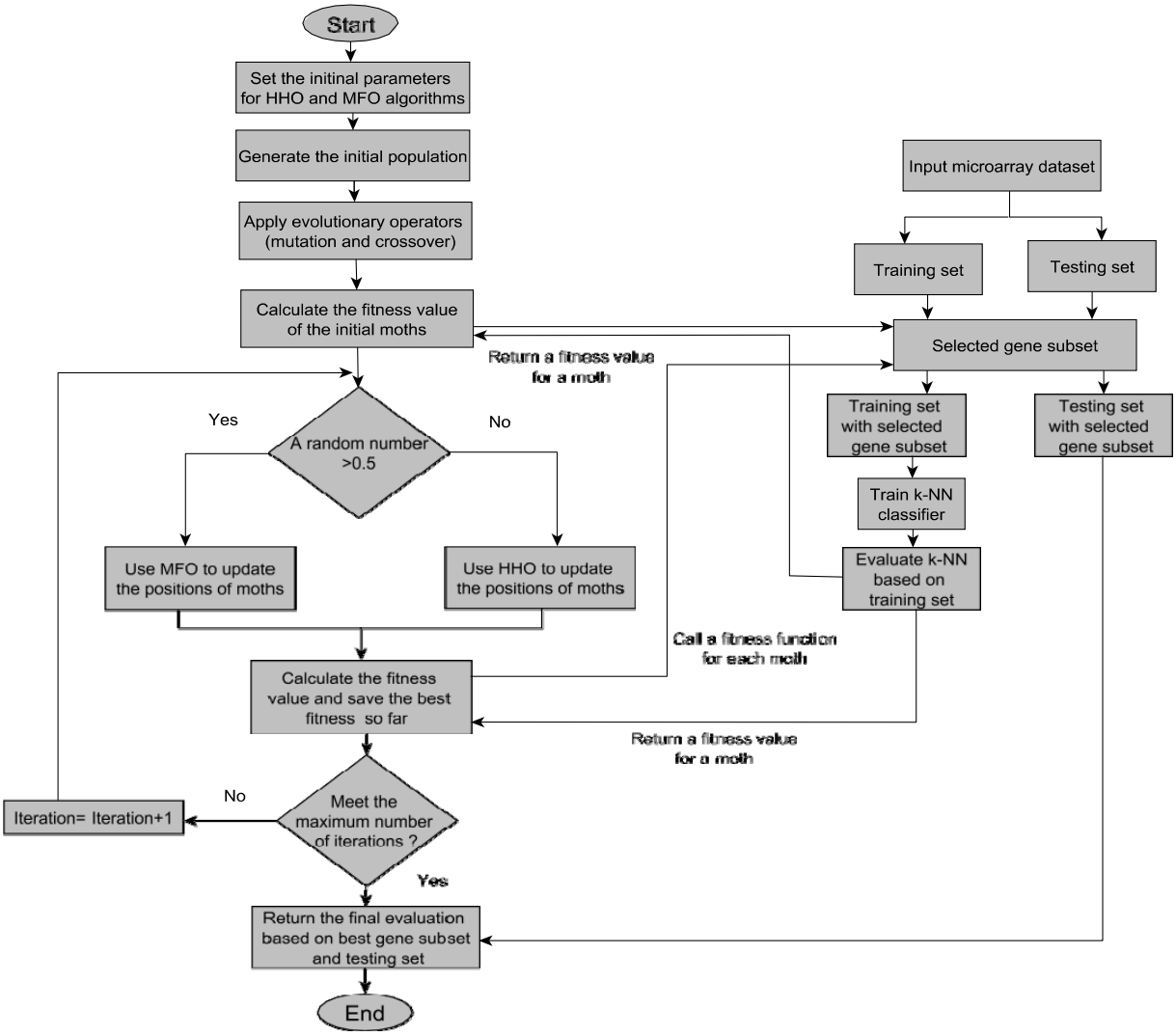

Fig. 3: Flowchart of the proposed wrapper approach (IHHOMFO) for genes selection.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

As mentioned earlier, in this chapter we intend to hybridize the two swarm-based systems MFO and HHO in one model called MFOHHO. The main objective is to complement the features of both algorithms for tackling the GS problem and enhance the classification performance of diseases. The update strategy of the HHO is merged into the structure of MFO to improve the updating of the positions of the moths in the gene space. The goals of this merging are adding more flexibility to the MFO, enhancing its capability in exploring the gene space, increasing the diversity of the positions of moths, and reaching the global optimal solution quickly. The steps of the proposed GS algorithm are shown in Fig. 3. The first step is to set the parameters of the MFO and HHO algorithms. Then, generate the initial population that represents the candidate solutions for the GS problem. The generated population is then modified by applying the crossover and mutation evolutionary operators to increase the diversity among the moths (these are explained in Subsection 3.2). In the next step, the fitness value of the moths is computed and the best moth is determined. Next, the MFOHHO updates the moths by using either MFO or HHO based on the quality of the fitness function (measured by its probability). If the probability of the fitness function for the current moth is larger than 0.5 then the MFO is applied, otherwise, the HHO is applied. The probability of the fitness function is computed as illustrated in Eq 17. Based on the value of *Proi* the current moth (e.g xi) is updated by either MFO or HHO.

$$Pro_i = \frac{f_i}{\sum_{i=1}^{n} f_i} (17)$$

Next, the fitness value is computed for each moth using an internal training and testing methodology (internal evaluation). As well as, the best moth from the updated population is determined. After that, a stopping condition is checked (e.g maximum number of iterations is reached). If it is satisfied, then the best gene subset is evaluated on the hidden testing part of the gene data set (external evaluation). If the stopping condition is not met then the steps from computing the probability of the current moth to the end are repeated. If the algorithm has executed a certain number of iterations, then the process halts and outputs the best gene subset encountered. The best gene subset found is then output as the recommended set of genes to be used in the actual design of the classification system. The gene search space is examined by two different search engines. These two swarm-based methodologies are used to update the positions of moths. Therefore, MFO can utilize HHO's merits and operations to support the exploration phase of the gene space. This leads to a better trade-off between exploration and exploitation.

3.2 Improved MFOHHO (IMFOHHO) Algorithm

In this approach, the crossover and mutation evolutionary operators are used to generate the diversified initial positions. The prime advantage of using crossover and mutation is to increase the diversity and randomness in the initialization process. Crossover swaps the bits after a random cutting point between the parents to generate new offsprings. Eq18 shows how a crossover operator is used to combine solutions, where on is an operator that performs the crossover on the two binary solutions X_i and X_{i-1}. In Eq19, the crossover switches between two input solutions with the same probability .X_d is the value of the *dth* dimension
in the obtained solution after applying the crossover operator on *Xi* and X_{i-1}.

$$X_i^{t+1} = \bowtie (X_i, X_{i-1}) \quad (18)$$

$$X^d = \begin{cases} X_1^d & if\ rand \geq .5 \\ X_2^d & otherwise \end{cases} \quad (19)$$

Mutation alters one or more gene values to change the solution from its previous state. Eq20 identifies the mutation process where $X_i^d(t+1)$ is the *ith* element at *dth* dimension in $X_i$solution,

$$X_i^d(t+1) = \begin{cases} 0 & if\ rand \geq .5 \\ 1 & otherwise \end{cases} \quad (20)$$

The steps of generating initial positions of the population are described below.
1. Initialization. Solutions are randomly generated.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

2. Crossover. A single point crossover method is used to create offspring solutions.
3. Mutation. Uniform mutation is adopted.
4. Decode. Decode mutated Solutions as the initial positions of the population.

3.3 Time Complexity of the Proposed IMFOHHO

The computational complexity of the MFO algorithm mainly depends on the number of moths, number of dimensions, maximum number of iterations, and the sorting mechanism of flames in each iteration. Since we use the Quick sort algorithm, the run time complexity is *O(nlogn)* and $O(n^2)$ in the average and the worst case, respectively. Considering the P function, the overall computational complexity is defined as follows:

$O(MFO)$=O(t(O(Quicksort)+O(positionupdate)))
$O(MFO)$=O(t(n^2+nd))=O(tn^2 +tnd)

Where n is the number of moths, t is the maximum number of iterations, and d is the number of dimensions.
The computational complexity of the HHO mainly depends on three processes: initialization, fitness evaluation, and updating of hawks. Note that within hawks, the computational run time complexity of the initialization process is *O(n).* The computational complexity of the updating mechanism is *O(tn) +O(tnd),* which is composed of searching for the best location and updating the location vector of all hawks, where t is the maximum number of iterations and *d* is the dimension of specific problems. Therefore, computational complexity of HHO is $O(n \times (t + td + 1))$.
The overall run time complexity of the IMFOHHO algorithm in the worstcase equals the higher complexity of MFO and HHO algorithms in the worst case.
The MFO algorithm has a higher complexity than HHO because the Quick sort run time complexity is $O(n^2)$ in the worst case. Thus, the overall run time complexity of the IMFOHHO equals the complexity of the MFO algorithm in the worst case. This means that there is no additional complexity.
O(IMFOHHO)=O(t(n^2+ n^d)) =O(tn^2+ tn^d).

3.4 The binary IMFOHHO with Transfer Function

GS is a binary search problem. Optimizing in a binary search space requires modifying the continuous optimizer by using binary solutions and binary update process. The gene solutions space is composed of binary gene subsets that represent the possible solutions for the GS problem. Each gene subset is composed of genes that have two binary values either "zero" or "one". The value of "zero" means that the corresponding gene is not selected. The value of 'one' means that the corresponding gene is selected. A GS problem is a switching of some elements of the solutions to update their positions in the gene space until they approach the global optimal solution. A binary update process depends on a way to define when the gene has to be converted to "zero" or "one." A mapping procedure is used to map the continuous variables of the original continuous optimizer into binary variables. Transfer functions are efficient methods that are commonly used to generate binary variants.

In [35], Mirjalili and Lewis used these transfer functions for binary conversion of the PSO algorithm. They define a probability of updating an element of a solution (switching between "zero" and "one"). In [36], Kennedy and Eberhart designed a binary version of PSO algorithm using the sigmoid function as in Eq.21 where $X_i^j(t)$ indicates the velocity of particle i at dimension j in iteration *t.* All velocity values were converted to probability values within a range [0,1].

$$T(X_i^j(t)) = 1/(1 + e^{-X_i^j(t)}) \quad (21)$$

After defining a probability for each element of the position vector, it can be updated by S-shaped TFs as illustrated in Eq.22 where $X_i^j(t+1)$represents the *ith* element in the X solution at dimension d in iteration *t+1*, rand ∈ [0; 1], which was generated using a random probability distribution.

$$X_i^j(t+1) = \begin{cases} 0 & if\ rand < T(X_i^j(t+1)) \\ 1 & otherwise \end{cases} \quad (22)$$

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

3.5 BIMFOHHO for Gene Selection

In this chapter, the k-Nearest Neighbor (KNN) classifier is used to evaluate each feature subset. The target of the FS is to simultaneously achieve two contradictory objectives, which are maximizing the classification accuracy and minimizing the number of features. In this work, the fitness function in Eq.23 is used to evaluate the selected subsets in all approaches where $\alpha\gamma_R(D)$ is the error rate, |R| is the number of selected features in the reduced data set, and |C| is the number of features in the original data set, and $\alpha \in [0,1]$, $\beta = (1-\alpha)$are two parameters that indicate the importance of classification and length of feature subset according to recommendations [6].

$$Fitness = \alpha\gamma_R(D) + \beta\frac{|R|}{|C|} \quad (23)$$

Table 1: Gene Expression Datasets Characteristics

Datasets	# samples	# Genes	#classes
Breast	97	24,481	2
Mll	72	12,582	3
Colon	62	2000	2
All-AML	72	7129	2
ALL-AML-3C	72	7129	3
ALL-AML-4C	72	7129	4
CNS	60	7129	2
Ovarian	253	15,154	2
SRBCT	83	2308	4
Lymphoma	62	4026	3

4 EXPERIMENTAL SETUP AND RESULTS

4.1 Machine Specification

All the experiments were executed on a personal machine with AMD AthlonDual-Core QL-60 CPU at 1.90 GHz and memory of 2 GB running Windows7Ultimate 64 bit operating system. The optimization algorithms have been all implemented in Python in the EvoloPy-FS framework [6].

4.2 Dataset Used

In this chapter, ten microarray datasets from 1 are used to evaluate the performance of the proposed approach. The selected benchmark datasets include"Breast", "MLL","Colon", "ALLAML", "ALLAML-3C", "ALLAML-4C", "Lung","CNS", "Ovarian" and "SRBCT" data sets. These data sets are commonly used in many studies and cover the example of small, medium, and large dimensional data sets. The characteristics of the selected data sets are summarized in Table1. Table 1 contains the number of genes (# Genes), the number of samples of each dataset (# Samples), and the number of classes (# Classes).

4.3 Parameter Settings

The maximum number of iterations and the population size were set to 100and 10, respectively. In this work, the K-NN classifier (K = 5) is used to assess the goodness of each solution in the wrapper FS approach. Each dataset is randomly divided into two parts; 80% for training and 20% for testing. To obtain statistically significant results, this division was repeated 30 independent times. Therefore, the final statistical results were obtained over 30 independent runs. The BBA and BCS are used for comparison with the proposed approaches. The BBA parameters are *QminFrequencyminimum = 0*, *QmaxFrequencymaximum = 2*, *ALoudness* = 0,5, *rPulserate = 0,5* The BCS parameters are *pa = 0,25* and $\beta = 3/2$.

4.4 Performance Measures

The used evaluation measures are fitness values (see Eq. 24), classification accuracy (see Eq. 25), number of selected features (see Eq. 26) and CPU time (see Eq. 27).

$\frac{1}{M}\sum_{j=1}^{M} Fit^i$ (24)

where M is the number of runs, Fit^i the fitness value of the best solution in the *ith* run.

$$\frac{1}{M}\sum_{j=1}^{M}\frac{1}{N}\sum_{i=1}^{N} C_i = L_i \quad (25)$$

where *M* is the number of runs for MFO to find the optimal subset of features, *N* is the number of dataset instances, C_i is the predictive class, and L_i is the actual class.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

$$\frac{1}{M}\sum_{i=1}^{M}\frac{d_i}{D} \qquad (26)$$

Where M is the number of runs, d_i is the number of selected features in the best solution from the i_{th} run, and D is the total number of features in the original dataset.

$$\frac{1}{M}\sum_{i=1}^{M} RT_i \qquad (27)$$

Where M is the number of runs, RT_i is the running time required for the i_{th} run to be completed.

4.5 Experimental Results and Discussion

This subsection studies the performance of the proposed approach (BIMFOHHO) over the ten microarray data sets described in Table 1. The BIMFOHHO is compared against the standard algorithms MFO and HHO, then it is compared with two well-known SI algorithms that are used as wrapper-GS approaches namely BBA and BCS. All the experiments have been conducted using the same machine specification, with the same settings of parameters and run in the same software framework, which is EvoloPy-FS.

The results of the proposed BIMFOHHO approach with the other approaches are summarized in Table 2. The comparisons are based on three evaluation measures the average accuracy, the average fitness, the average number of selected genes, and the average running times (in seconds). From Table 2, BIMFOHHO achieved superior classification results across 80% of the data sets. The BHHO achieved the highest performance on ALL-AML4c and the BCS achieved the highest performance on Lymphoma. In terms of fitness values, BIMFOHHO achieved the minimum fitness value across five data sets, namely Breast, MILL, Colon, Ovarian, and SRBCT. The BMFO achieved the minimum fitness on ALL-AML and ALL-AML3c data sets. BHHO achieved the minimum fitness on ALL-AML4c and BCS achieved the minimum on CNS and Lymphoma. Based on the number of selected genes, BIMFOHHO achieved the best reduction rate across all the data sets except ALL-AML3c and ALL-AML4c, it came in the second rank. For the running time, BIMFOHHO achieved the lowest computation time across 50% of the data sets. It achieved second-lowest computation time on MLL and Colon. According to fitness values, BIMFOHHO achieved the lowest fitness value across three data sets namely Breast, CNS, and Ovarian. By comparing the running time of BIMFOHHO, BHHO, and BMFO, it can be seen that BIMFOHHO achieved the lowest computation time across seven data sets while the BMFO achieved the lowest computation time across the remaining three datasets namely, ALL-AML, SRBCT, and Lymphoma. The BHHO was not better than BIMFOHHO and BMFO in terms of running time across any of the data sets.

5 CONCLUSION AND FUTURE WORK

This chapter introduces a new MH-MH wrapper approach to address the GS problem. In the proposed approach, the HHO and MFO algorithms are combined to effectively explore the gene space using different mathematical models. The MFOHHO model uses the best features of HHO to improve the capability of MFO as a search engine in the GS process. To make a broad coverage of the gene search space, the initialization procedure of MFOHHO is improved by using a set of evolutionary operators (i.e crossover and mutation). These operators can achieve more randomness among the individuals of the population. The evaluation process is performed by comparing the proposed IMFOHHO with the original MFO and HHO algorithms. It is then compared with three well-known wrapper-based GS approaches using several evaluation measurements namely the average accuracy, the average fitness, the average number of selected genes, and the average running times (in seconds). BIMFOHHO achieved superior classification accuracy results and gene selection rate across 80% of the data sets and achieved the best fitness values and running time across 50% of the datasets. The incorporation of multiple modifications on the standard MFO algorithm results in an efficient GS tool that can identify the most biologically relevant genes related to cancer. For future work, the performance of the proposed approach can be investigated in other applications such as hyperspectral image processing, face recognition, an electroencephalogram (EEG) application. As well as, the algorithm can be improved to be multi-objective.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 2: The Results Of The Classifier To Evaluate The Selected Genes Over All Datasets

Algorithm		Dataset				
		Breast	MLL	Colon	ALL-AML	ALL-AML3c
BMFO	#A	75.800	95.341	79.391	85.822	83.140
	#F	0.526	0.997	0.220	0.171	0.337
	#G	15537.800	8757.700	1250.800	4498.233	4847.267
	#T	10536.589	4298.182	762.211	2640.002	2624.0212
BHHO	#A	75.758	95.331	79.395	85.830	83.153
	#F	0.647	0.972	0.203	0.009	0.252
	#G	15545.859	8762.981	1254.880	4502.112	4854.859
	#T	10587.981	4717.055	769.100	2696.000	2742.019
BIMFOHHO	#A	78.806	95.403	79.422	85.846	83.155
	#F	0.244	0.954	0.101	0.127	0.337
	#G	15534.229	8751.698	1241.267	4495.058	4849.354
	#T	10535.002	4296.058	760.055	2663.002	2621.195
BCS	#A	75.730	95.318	79.395	85.830	83.153
	#F	0.695	0.998	0.159	0.186	0.295
	#G	15538.873	8765.652	1249.320	4500.112	4850.447
	#T	10544.591	4299.180	765.991	2638.569	2625.196
BBA	#A	75.721	95.306	79.303	85.202	83.149
	#F	0.708	0.999	0.237	0.474	0.315
	#G	15562.354	8777.732	1267.224	4511.878	4877.250
	#T	10540.991	4296.001	758.331	2636.142	2621.258
		ALL-AML4c	CNS	Ovarian	SRBCT	Lymphoma
BMFO	#A	81.425	66.456	91.623	87.623	98.315
	#F	0.337	0.205	0.171	0.343	0.187
	#G	4802.033	4654.200	9765.800	1657.233	2519.067
	#T	10536.589	4298.182	762.211	1819.014	2592.141
BHHO	#A	81.450	66.487	91.647	87.628	98.325
	#F	0.111	0.158	0.158	0.330	0.150
	#G	4814.290	4662.254	9772.899	1666.274	2529.148
	#T	10533.026	4300.897	775.889	2640.002	2624.021
BIHHOMFO	#A	81.443	66.510	91.648	87.640	98.321
	#F	0.227	0.202	0.125	0.328	0.174
	#G	4810.852	4647.421	9744.542	1636.579	2509.009
	#T	10535.227	4293.115	761.104	2123.000	2621.188
	#T	10535.227	4293.115	761.104	2123.000	2621.188
BCS	#A	81.431	66.487	91.647	87.628	98.416
	#F	0.295	0.068	0.242	0.339	0.013
	#G	4814.556	4662.287	9773.999	1666.589	2529.451
	#T	10537.897	43001.859	767.281	2225.142	2633.748
BBA	#A	81.434	66.432	91.604	87.623	98.323
	#F	0.229	0.312	0.198	0.343	0.175
	#G	4843.589	4673.589	9776.882	1668.100	2539.777
	#T	10533.143	4300.189	765.449	2000.780	2588.784

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

REFERENCES

[1] Heller, Michael J. "DNA microarray technology: devices, systems, and applications." Annual review of biomedical engineering 4.1 (2002): 129-153.

[2] Yang, Cheng-Huei, Li-Yeh Chuang, and C. Hong Yang. "IG-GA: a hybrid filter/wrapper method for feature selection of microarray data." Journal of Medical and Biological Engineering 30.1 (2010): 23-28.

[3] Alomari, Osama Ahmad, et al. "A hybrid filter-wrapper gene selection method for cancer classification." 2018 2nd International Conference on BioSignal Analysis, Processing and Systems (ICBAPS). IEEE, 2018.

[4] Chuang, Li-Yeh, et al. "Gene selection and classification using Taguchi chaotic binary particle swarm optimization." Expert Systems with Applications 38.10 (2011): 13367-13377.

[5] Mohamad, MohdSaberi, et al. "A modified binary particle swarm optimization for selecting the small subset of informative genes from gene expression data." IEEE Transactions on information Technology in Biomedicine 15.6 (2011): 813-822.

[6] Khurma, Ruba Abu, et al. "EvoloPy-FS: An Open-Source Nature-Inspired Optimization Framework in Python for Feature Selection." Evolutionary Machine Learn-ing Techniques. Springer, Singapore, 2020. 131-173.

[7] Chuang, Li-Yeh, et al. "Improved binary PSO for feature selection using gene expression data." Computational Biology and Chemistry 32.1 (2008): 29-38.

[8] Tran, Binh, Bing Xue, and Mengjie Zhang. "Improved PSO for feature selection on high-dimensional datasets." Asia-Pacific Conference on Simulated Evolution and Learning. Springer, Cham, 2014.

[9] Emary, Eid, Hossam M. Zawbaa, and Aboul Ella Hassanien. "Binary grey wolf optimization approaches for feature selection." Neurocomputing 172 (2016): 371-381.

[10] Yang, Cheng-San, et al. "A Hybrid Feature Selection Method for Microarray Classification." IAENG International Journal of Computer Science 35.3 (2008).

[11] Martinez, Emmanuel, Mario Moises Alvarez, and Victor Trevino. "Compact cancer biomarkers discovery using a swarm intelligence feature selection algorithm." Computational biology and chemistry 34.4 (2010): 244-250.

[12] Chuang, Li-Yeh, et al. "Gene selection and classification using Taguchi chaotic binary particle swarm optimization." Expert Systems
with Applications 38.10 (2011): 13367-13377.

[13] Mohamad, MohdSaberi, et al. "A modified binary particle swarm optimization for selecting the small subset of informative genes from gene expression data." IEEE Transactions on information Technology in Biomedicine 15.6 (2011): 813-822.

[14] Banka, Haider, and Suresh Dara. "A Hamming distance based binary particle swarm optimization (HDBPSO) algorithm for high dimensional feature selection, classification and validation." Pattern Recognition Letters 52 (2015): 94-100.

[15] Xi, Maolong, et al. "Cancer feature selection and classification using a binary quantum-behaved particle swarm optimization and support vector machine." Computational and mathematical Methods in Medicine 2016 (2016).

[16] Zawbaa, Hossam M., et al. "Large-dimensionality small-instance set feature selec- tion: A hybrid bio-inspired heuristic approach." Swarm and Evolutionary Computation 42 (2018): 29-42.

[17] Moghadasian, Mahmood, and SeyedehParvaneh Hosseini. "Binary cuckoo opti- mization algorithm for feature selection in high-dimensional datasets." International conference on innovative engineering technologies (ICIET'2014). 2014.

[18] Ibrahim, Hadeel Tariq, et al. "A grasshopper optimizer approach for feature selection and optimizing SVM parameters utilizing real biomedical data sets." Neural Computing and Applications 31.10 (2019): 5965-5974.

[19] Robbins, K. R., et al. "The ant colony algorithm for feature selection in high-dimension gene expression data for disease classification." Mathematical Medicine and Biology: a Journal of the IMA 24.4 (2007): 413-426.

[20] Kabir, Md Monirul, Md Shahjahan, and Kazuyuki Murase. "A new hybrid ant

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

colony optimization algorithm for feature selection." Expert Systems with Applications 39.3 (2012): 3747-3763.

[21] Kabir, Md Monirul, Md Shahjahan, and Kazuyuki Murase. "A new local search based hybrid genetic algorithm for feature selection." Neurocomputing 74.17 (2011): 2914-2928.

[22] Frohlich, Holger, Olivier Chapelle, and Bernhard Scholkopf. "Feature selection for support vector machines by means of genetic algorithm." Proceedings. 15th IEEE International Conference on Tools with Artificial Intelligence. IEEE, 2003.

[23] Huang, Jinjie, Yunze Cai, and Xiaoming Xu. "A hybrid genetic algorithm for feature selection wrapper based on mutual information." Pattern Recognition Letters 28.13 (2007): 1825-1844.

[24] Lu, Jianjiang, Tianzhong Zhao, and Yafei Zhang. "Feature selection based-on genetic algorithm for image annotation." Knowledge-Based Systems 21.8 (2008): 887- 891.

[25] Mirjalili, Seyedali. "Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm." Knowledge-based systems 89 (2015): 228-249.

[26] Sayed, Gehad Ismail, et al. "Alzheimer's disease diagnosis based on moth flame optimization." International Conference on Genetic and Evolutionary Computing. Springer, Cham, 2016.

[27] Wang, Mingjing, et al. "Toward an optimal kernel extreme learning machine using a chaotic moth-flame optimization strategy with applications in medical diagnoses." Neurocomputing 267 (2017): 69-84.

[28] Heidari, Ali Asghar, et al. "Harris hawks optimization: Algorithm and applications." Future generation computer systems 97 (2019): 849-872.

[29] Wang, Qiwei, Guangliang Ren, and Jing Tu. "A soft handover algorithm for TD-LTE system in high-speed railway scenario." 2011 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC). IEEE, 2011.

[30] Liu, Min, et al. "Design and evaluation of vertical handoff decision algorithm in heterogeneous wireless networks." 2006 14th IEEE International Conference on Net- works. Vol. 2. IEEE, 2006.

[31] Liu, Min, et al. "Design and evaluation of vertical handoff decision algorithm in heterogeneous wireless networks." 2006 14th IEEE International Conference on Net- works. Vol. 2. IEEE, 2006.

[32] Liu, Min, et al. "Design and evaluation of vertical handoff decision algorithm in heterogeneous wireless networks." 2006 14th IEEE International Conference on Net- works. Vol. 2. IEEE, 2006.

[33] Thaher, Thaer, et al. "Binary Harris Hawks Optimizer for High-Dimensional, Low Sample Size Feature Selection." Evolutionary Machine Learning Techniques. Springer, Singapore, 2020. 251-272.

[34] Yang, Xin-She. Nature-inspired metaheuristic algorithms. Luniver press, 2010.

[35] Mirjalili, Seyedali, and Andrew Lewis. "S-shaped versus V-shaped transfer functions for binary particle swarm optimization." Swarm and Evolutionary Computation 9 (2013): 1-14.

[36] Kennedy, James, and Russell C. Eberhart. "A discrete binary version of the particle swarm algorithm." 1997 IEEE International conference on systems, man, and cybernetics. Computational cybernetics and simulation. Vol. 5. IEEE, 1997.

[37] Khurma, R.; Aljarah, I. and Sharieh, A. (2020). An Efficient Moth Flame Optimization Algorithm using Chaotic Maps for Feature Selection in the Medical Applications. In Proceedings of the 9th International Conference on Pattern Recognition Applications and Methods - Volume 1: ICPRAM, ISBN 978-989-758-397-1, ISSN 2184-4313, pages 175-182. DOI: 10.5220/0008960701750182.

[38]Khurma, Ruba Abu, Ibrahim Aljarah, and Ahmad Sharieh. "Rank Based Moth Flame Optimization for Feature Selection in the Medical Application." 2020 IEEE Congress on Evolutionary Computation (CEC). IEEE, 2020.

[39] Al-Betar, Mohammed Azmi, Osama Ahmad Alomari, and Saeid M. Abu-Romman. "A TRIZ-inspired bat algorithm for gene selection in cancer classification." Genomics 112.1 (2020): 114-126.