

Personal Explanation:

PO-Algorithm:

**In Party Formation and Constituency Allocation**, The total population P is distributed into n political parties. Each party 𝑃𝑖 has n members in it. The jth member of an ith party is represented as 𝑝𝑖 𝑗 . The members are represented by using a d-dimensional vector which is the number of input variables. The kth dimension is represented as 𝑝𝑖,𝑘 𝑗 which means jth member of ith political party in the kth dimension. The constituencies are divided into n different constituencies. The jth member of each party competes for an election in the jth constituency. The set of selected party leaders and constituency winners are represented and these sets are calculated in the election phase.  
  
**In election campaign**, the member vector improves its chances of getting elected by improving based on pos based on RPPUS (past-based position updating strategy), pos based on leaders of parties and pos based on area/constituency winners.

**In partyswitching**, lamda starts from its max value to zero, each member is selected with a lamda proability and then switched with least member of a random party.

**In election**, fitness of each member for election in each area is evaluated and the best one is winner of that area/ constituency.

**In government formation/ Parliamentary Affairs**, each area winner and leaders of parties are selected. Each member is updated to randomly selected parliament member. If solution fitness improve then ok otherwise no change.

election campaign -> party switching -> election -> parliamentary affairs phase provide optimization by performing different local and global searches. The election campaign and parliamentary affairs provide exploitation by searching in the most promised regions of search space which allows the PO to show superior performance.  
  
BPO-Algorithm:  
?

HLBPO-Algorithm:  
This algorithm utilizes the concept of updating the position of candidate solutions to personal best and global best solutions. In the PO algorithm, the position of candidates was updated using the personal best solutions (party leaders) in election campaign middle method.

Algorithm 1: Hyper Learning Binary Political Optimizer Algorithm (HLBPO)

Input:

n (number of party members, political parties and constituencies),

λ𝑚𝑎𝑥 (The upper limit of the party-switching rate),

𝑇𝑚𝑎𝑥(total number of iterations),

plr (personal learning rate),

glr (global learning rate)

Output: final population P(𝑇𝑚𝑎𝑥)

Party Formation and Constituency Allocation (P, n)

Initialization (n)

Election (n)

t = 1;

λ = λ𝑚𝑎𝑥;

while t ≤ 𝑇𝑚𝑎𝑥 do

P(t − 1) = P;

f (P(t − 1)) = f (P);

foreach Pi ∈ P do

foreach 𝑝𝑖 𝑗 ∈ Pi do

𝑝𝑖 𝑗 = ElectionCampaign(𝑝𝑖 𝑗 , 𝑝𝑖 𝑗 (t − 1), 𝑝𝑖 ∗ , 𝑐𝑗 ∗ );

end

end

PartySwitching (P, λ);

Election (n);

ParliamentaryAffairs (𝐶 ∗ , 𝑃);

Hyper Learning Position Updating Strategy (n, plr, glr)

λ = λ − λ𝑚𝑎𝑥⁄T𝑚𝑎𝑥;

t = t + 1;

end

**In initialization**, The HLBPO first sets the parameters and then initializes a random population of candidate solutions with values 0 and 1. If the value is 0 it means that the particular feature is not selected and if the corresponding value is 1 then the feature is selected

Algorithm 2: Initialization (n)

foreach Pi ∈ P do

foreach 𝑝𝑖 𝑗 ∈ Pi do

r ← random number in the interval [0, 1]

if r ≤ 0.5 then

𝑝𝑖 𝑗 = 0

else

𝑝𝑖 𝑗 = 1

end

end

end

Table 1 shows the parameter setting of HLBPO. The number of solutions is set to be 10 and the total iterations are set to be 100. Two new parameters are used namely plr and glr which need a very careful adjustment.

Parameters Values

Number of Candidate Solutions (N) 10

Total Iterations (T) 100

Dimensions (D) Number of features in each dataset

Global Learning Rate (glr) 0.7

Personal Learning Rate (plr) 0.4

**Hyper Learning Position Updating Strategy**

This strategy enables the solutions to learn from both personal best solutions as well as global best solutions in the course of their search phase. In the proposed technique, the position of candidates is updated using the following Eq. (13) and (14).

𝑝𝑖 𝑗 (𝑡 + 1) = { 𝑝𝑖 𝑗 0 ≤ 𝑟1 < plr

𝑝𝑏𝑖 𝑗 (t) plr ≤ 𝑟1 < glr

𝑔𝑏 (𝑡) glr ≤ 𝑟1 ≤ 1 (13)

𝑝𝑖 𝑗 = { 1− 𝑝𝑖 𝑗 (𝑡) 𝑟2 < P (𝑝𝑖 𝑗 (t))

𝑝𝑖 𝑗 (𝑡) 𝑟2 ≥ P (𝑝𝑖 𝑗 (t)) (14)

Here 𝑝𝑖 𝑗 represents the position of the ith member in jth political party, pb represents the position of the personal best candidate, gb is the position of the global best solution, t is the iteration number, 𝑟1 and 𝑟2 are two random numbers generated in the interval of [0,1] and plr and glr represents personal and global learning rates and their value is constant which lies in the interval of [0,1].

Hyper learning is shown in Algorithm 4.

Algorithm 4: Hyper Learning Position Updating Strategy (n, plr, glr)

foreach Pi ∈ P do

foreach 𝑝𝑖 𝑗 ∈ Pi do

TF = Calculate probability using Transfer Function in Eq. (16)

𝑟1 ← random number in the interval [0, 1]

if 0 ≤ 𝑟1 < plr then

Update position using Eq. (14)

elseif plr ≤ 𝑟1 < glr then

Update position with respect to personal best solution

elseif glr ≤ 𝑟1 ≤ 1

Update position with respect to global best solution

end

end

end

The plr and glr play a very significant role during the learning process.

If their values are kept too low then the search region will be around personal and global best solutions so the solutions are more prone to stuck in the local optima.

If the values of plr and glr are kept too high then the position updating strategy will become similar to the binary PO.

So the choice of these parameters is very important. Feature selection is a binary optimization problem. In the PO algorithm, the fitness of the members decides their positions in the search space but the fitness values are continuous which should be transformed into binary values to use for binary feature selection problem because in the feature selection technique the search space is a Boolean n-dimensional matrix.

To generate the probability of changing the position of candidate solutions, transfer functions are used.

There are two main categories of transfer function: S and V-shaped as shown in Figure 2 and Figure 3 [36]. Fig. 2. S-shaped Transfer Function [34] Fig. 3. V-shaped Transfer Function [35]

*In the first one, the political member’s goodwill is updated using the Sigmoid function (S-shaped) and called BPO-S. While, in the second one, we used the Hyperbolic Tangent transfer function, called BPO-V*

In this algorithm, the transfer function used to generate the probability is a V-shaped transfer function that computes the altering probability of the candidate’s position. This transfer function does not restrict the candidates to choose a value of either 0 or 1. This allows the algorithm to perform high exploration and find more promising regions in the search space.

The transfer function used in HLBPO is shown in Eq. (16).

𝑃 (𝑝𝑖 𝑗 (𝑡)) = 𝑇𝐹 (𝑝𝑖 𝑗 (𝑡)) (15)

𝑇𝐹(𝑥) = | 𝑥 √(𝑥^2+1) | (16)

.

**In partyswitching**, .

**In election**, the first fitness of each member is calculated using the fitness function. After calculating the fitness function, the sets of party leaders and constituency winners are formed. In the next step, the global best solution is recorded. The global best solution is the one whose fitness value is least among all the candidate solutions.

Algorithm 3: Election (n)

foreach Pi ∈ P do

foreach 𝑝𝑖 𝑗 ∈ Pi do

Calculate fitness of 𝑝𝑖 𝑗 using Eq. (18)

End

end

Compute set of party leaders 𝑃 ∗ using Eq. (12)

Compute set of constituency winners 𝐶 ∗ using Eq. (11)

Store the position and fitness of global best solution

End

**Fitness Evaluation:**

The fitness evaluation of all the candidate solutions will be done using an objective function. The purpose of using an objective function is to evaluate the quality of solutions. The objective function used in the proposed algorithm minimizes the features selection ratio and classification error rate. The objective function [8] is given in Eq. (18).

𝐹𝑖𝑡𝑛𝑒𝑠𝑠 = 𝛼𝐸𝑅 + 𝛽 ( |𝑆𝐹| / |𝑇𝐹| ) (18)

Here ER represents the error rate of classification in a specific classifier. The classifier used in this study is K nearest neighbor also known as the KNN classifier.

|SF| is the length of the subset of selected features and |TF| is the length of total features in the dataset. α and β are two parameters. α impacts the error of classification with a value in the range of [0, 1] and β impacts the feature size with a value of β = (1-α).

In the first step, the datasets are partitioned into two sets; training and validation sets with the help of a stratified 10 fold cross-validation technique. The training set is used to train the model and the validation set is the set of samples that are held back during the formation of the training set to evaluate the selected features. The reason for using the KNN algorithm is because it is simple and requires no prior training for making predictions so new data can be added very easily [37].

( Check the HLBDA code)

Step#5

Termination After every iteration, the position of the candidate solutions is updated. After a pre-defined number of iterations, the algorithm terminates.

New Personal Explanation:  
  
Political Optimizer (PO) Algorithm:  
It is a socio-inspired algorithm for solving the feature selection problem. This algorithm is inspired by the political system of our social environment. It incorporates different phases of politics. The exploration and exploitation are performed by using the recent past-based position updating strategy (RPPUS) which allows the algorithm to learn from the previous election. It learns by improving the weak performing solutions to better solutions like party leaders and by comparing the solutions with winners of constituencies and updating their positions to them. The balance in the exploration and exploitation is formed by the party-switching phase.   
Fitness evaluation of the candidate solutions is done using the election phase.   
To introduce exploitation and convergence in the algorithm, the parliamentary phase is incorporated.  
There are five major phases in this algorithm.   
  
**1. Party Formation and Constituency Allocation**The total population P is distributed into n political parties as expressed in Eq. (1). Each party Pi has n members in it as expressed in Eq. (2). The jth member of an ith party is represented as 𝑝𝑖 𝑗 . The members are represented by using a d-dimensional vector which is basically the number of input variables. The kth dimension is represented as 𝑝𝑖,𝑘 𝑗 which means jth member of i th political party in the k th dimension as expressed in Eq. (3).   
  
P = {P1, P2, P3, ... , Pn} Eq.(1)   
Pi = {𝑝𝑖 1 ,𝑝𝑖 2 ,𝑝𝑖 3 , … , 𝑝𝑖 𝑛 } Eq.(2)   
𝑝𝑖 𝑗 = {𝑝𝑖,1 𝑗 , 𝑝𝑖,2 𝑗 , 𝑝𝑖,3 𝑗 , … , 𝑝𝑖,𝑑 𝑗 }T Eq.(3)   
  
The constituencies are divided into n different constituencies as shown in Eq. (4). The jth member of each party competes for an election in the jth constituency as shown in Eq. (5).   
  
C = {C1, C2, C3, ... , Cn} Eq.(4)   
𝐶𝑗 = { 𝑝1 𝑗 , 𝑝2 𝑗 , 𝑝3 𝑗 , … , 𝑝𝑛 𝑗 } Eq.(5)   
  
The set of selected party leaders and constituency winners are represented as shown in Eq. (6) and Eq. (7). These sets are calculated in the election phase.   
  
P\* = {𝑝1 ∗ , 𝑝2 ∗ , 𝑝3 ∗ , … , 𝑝𝑛 ∗ } Eq.(6)   
C\* = {𝑐1 ∗ , 𝑐2 ∗ , 𝑐3 ∗ , … , 𝑐𝑛 ∗ } Eq.(7)   
  
**2. Election Campaign**In the election campaign, the solutions improve their chances of getting elected by improving themselves based on three aspects.   
First, the past learning experience is used to update the position of solutions by using RPPUS strategy as shown in Eq. (8) and Eq. (9).   
  
Second, the position of solutions is updated with respect to the leaders of their party.   
Finally, the position of candidate solutions is updated with respect to the constituency winners.   
This increases the chance of obtaining better solutions.   
The current fitness f (𝑝𝑖 𝑗 (t)) is compared to the previous fitness f (𝑝𝑖 𝑗 (t-1)) and if there is an improvement in the fitness, then Eq. (8) is used to update position otherwise Eq. (9) is used.   
  
In this equation, r symbolizes a random number in the range of 0 to 1 and m\* holds the kth dimension of both, party leader and constituency winner.   
Eq. (8):-   
𝑖𝑓 𝑝𝑖,𝑘 𝑗 (𝑡 − 1) ≤ 𝑝𝑖,𝑘 𝑗 (𝑡) ≤ 𝑚∗ 𝑜𝑟 𝑝𝑖,𝑘 𝑗 (𝑡 − 1) ≥ 𝑝𝑖,𝑘 𝑗 (𝑡) ≥ 𝑚∗ :   
 𝑝𝑖,𝑘 𝑗 (𝑡 + 1)= 𝑚∗ +𝑟 (𝑚∗ − 𝑝𝑖,𝑘 𝑗 (𝑡))  
𝑖𝑓 𝑝𝑖,𝑘 𝑗 (𝑡 − 1) ≤ 𝑚∗ ≤ 𝑝𝑖,𝑘 𝑗 (𝑡) 𝑜𝑟 𝑝𝑖,𝑘 𝑗 (𝑡 − 1) ≥ 𝑚∗ ≥ 𝑝𝑖,𝑘 𝑗 (𝑡) :  
 𝑝𝑖,𝑘 𝑗 (𝑡 + 1)= 𝑚∗ + (2𝑟 − 1)|𝑚∗ −𝑝𝑖,𝑘 𝑗 (𝑡)|   
𝑖𝑓 𝑚∗ ≤ 𝑝𝑖,𝑘 𝑗 (𝑡 − 1) ≤ 𝑝𝑖,𝑘 𝑗 (𝑡) 𝑜𝑟 𝑚∗ ≥ 𝑝𝑖,𝑘 𝑗 (𝑡 − 1) ≥ 𝑝𝑖,𝑘 𝑗 (𝑡) :  
 𝑝𝑖,𝑘 𝑗 (𝑡 + 1)= 𝑚∗ + (2𝑟 − 1)|𝑚∗ −𝑝𝑖,𝑘 𝑗 (𝑡 − 1)|   
  
Eq. (9):-   
𝑖𝑓 𝑝𝑖,𝑘 𝑗 (𝑡 − 1) ≤ 𝑝𝑖,𝑘 𝑗 (𝑡) ≤ 𝑚∗ 𝑜𝑟 𝑝𝑖,𝑘 𝑗 (𝑡 −1) ≥ 𝑝𝑖,𝑘 𝑗 (𝑡) ≥ 𝑚∗ :  
 𝑝𝑖,𝑘 𝑗 (𝑡 + 1)= 𝑚∗ +(2𝑟 − 1)|𝑚∗ − 𝑝𝑖,𝑘 𝑗 (𝑡)|   
𝑖𝑓 𝑝𝑖,𝑘 𝑗 (𝑡 − 1) ≤ 𝑚∗ ≤ 𝑝𝑖,𝑘 𝑗 (𝑡) 𝑜𝑟 𝑝𝑖,𝑘 𝑗 (𝑡 − 1) ≥ 𝑚∗ ≥ 𝑝𝑖,𝑘 𝑗 (𝑡) :  
 𝑝𝑖,𝑘 𝑗 (𝑡 + 1)= 𝑝𝑖,𝑘 𝑗 (𝑡 −1) + 𝑟 (𝑝𝑖,𝑘 𝑗 (𝑡) − 𝑝𝑖,𝑘 𝑗 (𝑡 − 1))   
𝑖𝑓 𝑚∗ ≤ 𝑝𝑖,𝑘 𝑗 (𝑡 − 1) ≤ 𝑝𝑖,𝑘 𝑗 (𝑡) 𝑜𝑟 𝑚∗ ≥ 𝑝𝑖,𝑘 𝑗 (𝑡 − 1) ≥ 𝑝𝑖,𝑘 𝑗 (𝑡) :  
 𝑝𝑖,𝑘 𝑗 (𝑡 + 1)= 𝑚∗ + (2𝑟 − 1)|𝑚∗ −𝑝𝑖,𝑘 𝑗 (𝑡 − 1)|   
  
**3. Party Switching**   
This phase is incorporated to balance out exploration as well as exploitation. There is a parameter named party switching rate represented as λ that starts form the value λmax and then decreases to zero. All the members 𝑝𝑖 𝑗 are selected with the probability of λ and then switched with the least fit member of a randomly selected party 𝑃𝑟 which is shown in Eq. (10).   
q = argmaxf 1≤𝑗≤𝑛 (𝑝𝑟 𝑗 ) Eq.(10)   
  
**4. Election**   
In this phase, the fitness of each member that is competing in the election for a constituency is evaluated and the fittest member is chosen as the winner of the constituency. In this phase, the fitness of each member that is competing in election for a constituency is evaluated and the fittest member is chosen as the winner of constituency as shown in Eq. (11).   
q = argminf 1≤𝑖≤𝑛 (𝑝𝑖 𝑗 ) Eq.(11)   
𝑐𝑗 ∗ = 𝑝𝑞 𝑗   
  
The party leaders are selected using Eq. (12). The fittest member of a party is chosen as its leader.   
q = argminf 1≤𝑗≤𝑛 (𝑝𝑖 𝑗 ) ∀i ∈ {1, . . . , n} Eq.(12)   
𝑝𝑖 ∗ = 𝑝𝑖 𝑞   
  
**5. Parliamentary Affairs**   
After the formation of the government, the winners of constituencies and the leaders of parties are selected. Then every member of the parliament is updated to some other randomly selected parliament member. If there is an improvement in the solution’s fitness, the solution is updated otherwise the solution is not updated and remains the same.   
  
The election campaign, party switching, election, and parliamentary affairs phase provide optimization by performing different local and global searches. The election campaign and parliamentary affairs provide exploitation by searching in the most promised regions of search space which allows the PO to show superior performance. PO algorithm is used to solve continuous problems.   
  
**BPO**:  
??

**Hyper Learning BASED Binary Political Optimizer Algorithm**   
  
In this article, a Hyper Learning-Based Binary Political Optimizer Algorithm (HLBPO) is introduced to solve the feature selection problem. This algorithm utilizes the concept of updating the position of candidate solutions to personal best and global best solutions. In the PO algorithm, the position of candidates was updated using the personal best solutions (party leaders). By introducing the concept of position updating to both personal best solution and global best solution, the solutions, and their searching behavior are expected to improve. The proposed HLBPO algorithm is shown:   
  
Algorithm 1: Hyper Learning Binary Political Optimizer Algorithm (HLBPO)   
Input:   
n (number of party members, political parties and constituencies),   
λ𝑚𝑎𝑥 (The upper limit of the party-switching rate),   
𝑇𝑚𝑎𝑥(total number of iterations),   
plr (personal learning rate),   
glr (global learning rate)   
  
Output: final population P(𝑇𝑚𝑎𝑥)   
  
Implementation:  
Party Formation and Constituency Allocation (P, n)   
Initialization (n)   
Election (n)   
t = 1;   
λ = λ𝑚𝑎𝑥;   
while t ≤ 𝑇𝑚𝑎𝑥 do   
 P(t − 1) = P;   
 f (P(t − 1)) = f (P);   
 foreach Pi ∈ P do   
 foreach 𝑝𝑖 𝑗 ∈ Pi do   
 𝑝𝑖 𝑗 = ElectionCampaign(𝑝𝑖 𝑗 , 𝑝𝑖 𝑗 (t − 1), 𝑝𝑖 ∗ , 𝑐𝑗 ∗ );   
 end

end   
 PartySwitching (P, λ);   
 Election (n);   
 ParliamentaryAffairs (𝐶 ∗ , 𝑃);   
 Hyper Learning Position Updating Strategy (n, plr, glr)   
 λ = λ − λ𝑚𝑎𝑥⁄T𝑚𝑎𝑥;   
 t = t + 1;   
end   
  
**Initialization**   
The HLBPO first sets the parameters and then initializes a random population of candidate solutions with values 0 and 1. If the value is 0 it means that the particular feature is not selected and if the corresponding value is 1 then the feature is selected.   
Algorithm 2: Initialization (n)   
Implementation:  
foreach Pi ∈ P do   
 foreach 𝑝𝑖 𝑗 ∈ Pi do   
 r ← random number in the interval [0, 1]   
 if r ≤ 0.5 then   
 𝑝𝑖 𝑗 = 0   
 else   
 𝑝𝑖 𝑗 = 1   
 end   
 end   
end   
  
Table 1 shows the parameter setting of HLBPO. The number of solutions is set to be 10 and the total iterations are set to be 100. Two new parameters are used namely plr and glr which need a very careful adjustment.

Parameters Values

Number of Candidate Solutions (N) 10

Total Iterations (T) 100

Dimensions (D) Number of features in each dataset

Global Learning Rate (glr) 0.7

Personal Learning Rate (plr) 0.4

**Election**   
In the election phase, the first fitness of each member is calculated using the fitness function. After calculating the fitness function, the sets of party leaders and constituency winners are formed. In the next step, the global best solution is recorded. The global best solution is the one whose fitness value is least among all the candidate solutions. The election phase steps are shown:   
  
Algorithm 3: Election (n)   
Implementation:  
foreach Pi ∈ P do   
 foreach 𝑝𝑖 𝑗 ∈ Pi do   
 Calculate fitness of 𝑝𝑖 𝑗 using Eq. (18)   
 end   
end   
Compute set of party leaders 𝑃 ∗ using Eq. (12)   
Compute set of constituency winners 𝐶 ∗ using Eq. (11)   
Store the position and fitness of global best solution   
End   
  
**Hyper Learning Position Updating Strategy**   
This strategy enables the solutions to learn from both personal best solutions as well as global best solutions in the course of their search phase. In the proposed technique, the position of candidates is updated using the following Eq. (13) and (14).   
  
𝑝𝑖 𝑗 (𝑡 + 1) =   
{ 𝑝𝑖 𝑗 0 ≤ 𝑟1 < plr   
 𝑝𝑏𝑖 𝑗 (t) plr ≤ 𝑟1 < glr   
 𝑔𝑏 (𝑡) glr ≤ 𝑟1 ≤ 1 Eq.(13)   
  
𝑝𝑖 𝑗 =   
{ 1− 𝑝𝑖 𝑗 (𝑡) 𝑟2 < P (𝑝𝑖 𝑗 (t))  
 𝑝𝑖 𝑗 (𝑡) 𝑟2 ≥ P (𝑝𝑖 𝑗 (t)) Eq.(14)   
  
Here 𝑝𝑖 𝑗 represents the position of the ith member in jth political party, pb represents the position of the personal best candidate, gb is the position of the global best solution, t is the iteration number, 𝑟1 and 𝑟2 are two random numbers generated in the interval of [0,1] and plr and glr represents personal and global learning rates and their value is constant which lies in the interval of [0,1].   
Hyper learning is shown:   
  
Algorithm 4: Hyper Learning Position Updating Strategy (n, plr, glr)   
Implementation:   
foreach Pi ∈ P do   
 foreach 𝑝𝑖 𝑗 ∈ Pi do   
 TF = Calculate probability using Transfer Function in Eq. (16)   
 𝑟1 ← random number in the interval [0, 1]   
 if 0 ≤ 𝑟1 < plr then   
 Update position using Eq. (14)   
 elseif plr ≤ 𝑟1 < glr then   
 Update position with respect to personal best solution   
 elseif glr ≤ 𝑟1 ≤ 1   
 Update position with respect to global best solution   
 end   
 end   
end   
  
The plr and glr play a very significant role during the learning process. If their values are kept too low, then the search region will be around personal and global best solutions, so the solutions are more prone to stuck in the local optima.   
If the values of plr and glr are kept too high, then the position updating strategy will become similar to the binary PO.   
So, the choice of these parameters is very important. Feature selection is a binary optimization problem. In the PO algorithm, the fitness of the members decides their positions in the search space, but the fitness values are continuous which should be transformed into binary values to use for binary feature selection problem because in the feature selection technique the search space is a Boolean n-dimensional matrix.   
To generate the probability of changing the position of candidate solutions, transfer functions are used. There are two main categories of transfer function: S and V-shaped   
 *In the first one, the political member’s goodwill is updated using the Sigmoid function (S-shaped) and called BPO-S. While, in the second one, we used the Hyperbolic Tangent transfer function, called BPO-V*  
  
In this algorithm, the transfer function used to generate the probability is a V-shaped transfer function that computes the altering probability of the candidate’s position.   
  
This transfer function does not restrict the candidates to choose a value of either 0 or 1. This allows the algorithm to perform high exploration and find more promising regions in the search space.   
  
The transfer function used in HLBPO is shown in Eq. (16).   
  
𝑃 (𝑝𝑖 𝑗 (𝑡)) = 𝑇𝐹 (𝑝𝑖 𝑗 (𝑡)) Eq. (15)   
𝑇𝐹(𝑥) = | 𝑥 /√𝑥^2+1 | Eq.(16)   
  
**Fitness Evaluation**   
The fitness evaluation of all the candidate solutions will be done using an objective function. The purpose of using an objective function is to evaluate the quality of solutions. The objective function used in the proposed algorithm minimizes the features selection ratio and classification error rate. The objective function [8] is given in Eq. (17).   
  
𝐹𝑖𝑡𝑛𝑒𝑠𝑠 = 𝛼𝐸𝑅 + 𝛽 ( |𝑆𝐹|/ |𝑇𝐹| ) Eq.(17)   
  
Here ER represents the error rate of classification in a specific classifier. The classifier used in this study is K nearest neighbor also known as the KNN classifier. |SF| is the length of the subset of selected features and |TF| is the length of total features in the dataset. α and β are two parameters. α impacts the error of classification with a value in the range of [0, 1] and β impacts the feature size with a value of β = (1-α). In the first step, the datasets are partitioned into two sets; training and validation sets with the help of a stratified 10-fold cross-validation technique. The training set is used to train the model and the validation set is the set of samples that are held back during the formation of the training set to evaluate the selected features. The reason for using the KNN algorithm is because it is simple and requires no prior training for making predictions so new data can be added very easily   
  
**Termination**   
After every iteration, the position of the candidate solutions is updated. After a pre-defined number of iterations, the algorithm terminates.