HYBRID BINARY DRAGONFLY ENHANCED PARTICLE SWARM OPTIMIZATION ALGORITHM FOR SOLVING FEATURE SELECTION PROBLEMS

MOHAMED A. TAWHID*  
Department of Mathematics and Statistics, Faculty of Science  
Thompson Rivers University, Kamloops, BC, V2C 0C8, Canada  
and  
Department of Mathematics and Computer Science, Faculty of Science  
Alexandria University, Moharam Bey 21511, Alexandria, Egyp

KEVIN B. DSOUZA  
Electrical and Computer Engineering  
The University of British Columbia  
Vancouver BC V6T 1Z4, Canada

(Communicated by Zhipeng Cai)

Abstract. In this paper, we present a new hybrid binary version of dragonfly and enhanced particle swarm optimization algorithm in order to solve feature selection problems. The proposed algorithm is called Hybrid Binary Dragonfly Enhanced Particle Swarm Optimization Algorithm (HBDESPO). In the proposed HBDESPO algorithm, we combine the dragonfly algorithm with its ability to encourage diverse solutions with its formation of static swarms and the enhanced version of the particle swarm optimization exploiting the data with its ability to converge to the best global solution in the search space. In order to investigate the general performance of the proposed HBDESPO algorithm, the proposed algorithm is compared with the original optimizers and other optimizers that have been used for feature selection in the past. Further, we use a set of assessment indicators to evaluate and compare the different optimizers over 20 standard data sets obtained from the UCI repository. Results prove the ability of the proposed HBDESPO algorithm to search the feature space for optimal feature combinations.

1. Introduction. Feature selection is a way for identifying the independent features and removing redundant ones from the dataset [6]. The objectives of feature selection are dimensionality reduction of the data, improving prediction accuracy, and understanding data for different machine learning applications [5]. In the real world, data representation often uses too many features with some redundant ones,
which means certain independent features can fill in for others and the dependent (redundant) features can be removed. Moreover, the output is influenced by the relevant features because they contain important information about the data and the results will be obscure if any of them is excluded [3]. The classical optimization techniques have some limitations in solving the feature selection problems [16] and hence evolutionary computation (EC) algorithms are the alternative for solving these limitations and searching for the optimum solution [4]. Evolutionary computation (EC) algorithms are inspired by nature, group dynamics, social behavior, and interaction of biological organisms in a group. The binary version of these algorithms allow us to investigate problems like feature selection and arrive at superior results.

Many heuristic algorithms have been used in an attempt to solve the feature selection problem. A survey on evolutionary computation approaches to feature selection is delineated in [4]. The authors in [2] used a firefly to solve feature selection problem. A binary bat for solving feature selection problem is shed light upon in [20]. [12] presents a feature subset selection approach by grey wolf optimization. Even hybrid algorithms have been used to solve feature selection problems. A hybrid genetic algorithm on mutual information is presented in [14].

The dragonfly algorithm (DA) is a novel swarm intelligence optimization technique [19]. The main inspiration of the DA algorithm originates from the static and dynamic swarming behaviors of dragonflies in nature. Two essential phases of optimization, exploration and exploitation, are designed by modelling the social interaction of dragonflies in navigating, foraging for food, and evading enemies when swarming statically or dynamically. The DA algorithm has been applied to discrete as well as multi-objective problems [19], but has not been applied to feature selection problems.

Particle swarm optimization (PSO) is a population based stochastic optimization technique developed by Eberhart and Kennedy in 1995 [11], inspired by social behavior of bird flocking or fish schooling. In past several years, even though PSO has been successfully applied in many research and application areas like the constrained non linear optimization problems [8], for optimal design of combinational logic circuits [9] also to real world hydraulic problems [17], there is little work in the domain of feature selection [1]. It is demonstrated that PSO gets better results in a faster, cheaper way compared with other methods. Another reason is that PSO is attractive there are few parameters to tweak. One version, with slight variations, works well in a wide variety of real world applications. Here an enhanced version of the standard PSO is used [18] to solve the feature selection problem.

Hybridization of different algorithmic concepts is a method to obtain better performing systems and is believed to benefit from synergy, i.e. usually it exploits and unites advantages of the individual pure strategies. It is mostly due to the no free lunch theorems [23] that the generalized view of metaheuristics changed and people recognized that there cannot exist a general optimization strategy which is globally better than any other. In fact, to solve a problem at hand most effectively, it almost always requires a specialized algorithm that needs to be compiled of adequate parts. Hybridization is classified into many categories [10], [22]. Hybridization of one metaheuristic with another is a popular method to enhance the performance of both the algorithms.

The aim of this work is to propose a new hybrid binary version of dragonfly and enhanced particle swarm optimization algorithm in order to solve feature selection
problems effectively. The hybridization allows us to combine the best aspects of both these algorithms and obtain better performance. In this paper, we propose a new hybrid algorithm, which is called HBDESPO Algorithm by combining the dragonfly algorithm with the enhanced particle swarm optimization algorithm in order to obtain superior results when we compare it to the respective individual algorithms. We test the binary HBDESPO algorithm on 20 standard datasets obtained from the UCI repository [13]. We use set of assessment indicators to evaluate and compare the different optimizers. Also, we compare the algorithm with the HBEPSOD, where the particle swarm optimization is carried out first and then given to the dragonfly algorithm. The experimental results show the ability of the proposed HBDESPO algorithm to search the feature space for optimal feature combinations.

The reminder of this paper is organized as follows. In Section 2, we present the definition of the feature selection problem. We summarize the main concepts of the dragonfly algorithm in Section 3. We present the main concepts of the enhanced particle swarm optimization algorithm in Section 4. In Section 5, we describe the main structure of the proposed HBDESPO algorithm. Section 6 provides details about the feature selection problem, evaluation criteria and an insight about the classifier used. In Section 7, we report the experimental results and finally, we give the conclusion Section 8.

2. Definition of the feature selection problem. In this section, we present the definition of the feature selection problem as follows. The feature selection problem can be defined as the selection of certain number of features out of the total number of available features in such a way that the classification performance is maximum and the number of selected features is minimum.

\[ \text{fitness} = \alpha \gamma_R(D) + \beta \frac{|C - R|}{|C|} \]  

(1)

Where \( \gamma_R(D) \) is the classification quality of set R relative to decision D, \( R \) is the length of selected feature subset, \( C \) is the total number of features, \( \alpha \) and \( \beta \) are two parameters corresponding to the importance of classification quality and subset length, \( \alpha \in [0,1] \) and \( \beta = 1 - \alpha \). The fitness function maximizes the classification quality; \( \gamma_R(D) \), and the ratio of the unselected features to the total number of features; \( \frac{|C - R|}{|C|} \). The above equation can be easily converted into a minimization problem by using error rate rather than classification quality and using selected features ratio rather than using unselected feature size. The minimization problem can be formulated as in Eq. 2.

\[ \text{fitness} = \alpha E_R(D) + \beta \frac{|R|}{|C|} \]  

(2)

where \( E_R(D) \) is the error rate of the classifier, \( R \) is the length of the selected feature subset and \( C \) is the total number of features. \( \alpha \in [0,1] \) and \( \beta = 1 - \alpha \) are constants used to control the weights of classification accuracy and feature reduction.

The intuition behind the fitness function is that we want the algorithm to choose features in such a way that classification accuracy is not compromised while maintaining the number of features to a minimum as is reflected in Eq. 2.
3. Overview of binary dragonfly algorithm. In the following subsection, we will give an overview of the main concepts and structure of the binary dragonfly algorithm as follows.

3.1. Main concepts and inspiration. The dragonfly algorithm derives its inspiration from the static and dynamic swarming behavior of dragonflies. These two behaviors are very similar to the two main phases of metaheuristic algorithms: exploration and exploitation. Dragonflies create sub swarms and fly over different areas in a static swarm, which is similar to the main objective of the exploration phase. In the dynamic swarm, however, dragonflies fly in bigger swarms and along one particular direction, which is favorable in the exploitation phase. The five concepts of Separation, Alignment, Cohesion, Attraction to food and Distraction from enemy are used to simulate the behavior of dragonflies in both static and dynamic swarms [19].

3.2. Definition of concepts.

1. Separation: This parameter controls separation between the different available solutions. It helps to explore the search space initially. The separation weight(s) assigned to the parameter controls this possibility.

\[ S = \sum_{k=1}^{N} x_i - \text{Neighbors}_{x_i}(k) \]  

where \( \text{Neighbors}_{x_i} \) is the matrix consisting of neighbors of \( x_i \) and \( N \) indicates the number of neighbors.

2. Alignment: This parameter dictates the alignment of a solution with the neighboring solutions. This parameter is given the alignment weight(a) and the arrival to the final solution is controlled through this weight.

\[ A = \frac{\sum_{k=1}^{N} \text{Neighbors}_{Deltax_i}(k)}{N} \]  

where \( \text{Neighbors}_{Deltax_i} \) is the matrix consisting of \( Deltax \) values of neighbors of \( i \). \( Deltax \) is the matrix that keeps track of change in position of the solutions.

3. Cohesion: This parameter represents the convergence of the solutions to a particular food source. It is defined as the distance between the mean of the neighboring solutions and the current solution. It is controlled by the cohesion weight(c).

\[ C_{\text{temp}} = \frac{\sum_{k=1}^{N} \text{Neighbors}_{x_i}(k)}{N} \]  

\[ C = C_{\text{temp}} - x_i. \]  

4. Attraction to food: As the name suggests this parameter determines the distance of the current solution from the best solution in the group(food source). The Food attraction weight(f) controls this parameter.

\[ F = \text{Food_pos} - x_i \]

where \( \text{Food_pos} \) is the best solution in the iteration.
5. **Distraction from enemy**: This parameter makes sure that the current solution stays away from bad solutions in the space. Its essentially the separation from the worst solution in the space and is controlled by the *enemy distraction weight* \( e \).

\[
E = \text{Enemy}_\text{pos} + x_i
\]

where \( \text{Enemy}_\text{pos} \) is the worst solution in the group.

### 3.3. Binary dragonfly algorithm

In this section, we present in details the main steps of the binary dragonfly algorithm as shown in Algorithm 1. The notations introduced majorly follow [19].

- **Step 1.** Initialize the values of swarm size \( SS(N) \), \( w_{\text{max}}, w_{\text{min}}, \Delta \text{t}_{\text{max}} \) and \( \text{max}_{\text{iter}} \).
- **Step 2.** Randomly initialize the population as \( x_i = (x_{i1}, x_{i2}, \ldots, x_{iD}) \in S \) for each solution and the \( \Delta \text{t} \) matrix where each element takes a binary value.
- **Step 3.** The following steps are repeated until the terminating criteria are met.
  - **Step 1.** Update the weights and the parameters as mentioned in the algorithm.
  - **Step 2.** The fitness value of each solution is calculated using \( f(x_i) \).
  - **Step 3.** The best solution \( \text{Food}_\text{pos} \) and the worst solution \( \text{Enemy}_\text{pos} \) are assigned.
  - **Step 4.** At each iteration \( t \), the \( \Delta \text{t} \) value of each particle is updated according to equation
    \[
    \Delta \text{t}(i,j) = w\Delta \text{t}(i,j) + sS(j) + aA(j) + cC(j) + fF(j) + eE(j).
    \]
  - **Step 4.** The continuous values are mapped to binary values using the V3 shaped transfer function
    \[
    T = \frac{|\Delta \text{t}(i,j)|}{\sqrt{(1+\Delta \text{t}(i,j)^2)}}
    \]
    and new solutions are generated.
- **Step 4.** Produce the global best as the best found solution.

### 4. Overview of binary enhanced particle swarm optimization

In the following section, we will give an overview of the main concepts and structure of the binary enhanced particle swarm optimization algorithm as follows.

#### 4.1. Main concepts and inspiration

The particle swarm optimization is a population based search method inspired from the swarm behavior (information interchange) of birds [15]. In PSO, initially a random population of particles is initialized and these particles move with certain velocity based on their interaction with other particles in the population. At each iteration the personal best achieved by each particle and the global best of all the particles is tracked and the velocity of all the particles is updated based on this information. Certain parameters are used to give weights to the global and personal best. In the enhanced version of the binary PSO [18], special type of S shaped transfer functions is used to convert a continuous value to a binary value instead of a simple hyperbolic tangent function.
Algorithm 1 Binary dragonfly algorithm

1: Set the initial value of swarm size $SS(N)$, $w_{\text{max}}$, $w_{\text{min}}$, $\Delta t_{\text{max}}$ and $\text{max}_{\text{iter}}$.
2: Randomly initialize the population as $x_i = (x_{i1}, x_{i2}, \ldots, x_{iD}) \in S$ for each solution and the $\Delta t_{\text{max}}$ matrix where each element takes a binary value.
3: Set $t := 0$. \{Counter initialization\}
4: $\text{temp} = 0$. \{Counter initialization\}
5: $w = w_{\text{max}} - t \left( \frac{w_{\text{max}} - w_{\text{min}}}{\text{max}_{\text{iter}}} \right)$
6: if $(\text{temp} < 0)$ then
7: $\text{temp} = 0$
8: end if
9: $s = 2 \cdot \text{rand}. \text{temp}$
10: $a = 2 \cdot \text{rand}. \text{temp}$
11: $c = 2 \cdot \text{rand}$
12: $f = 2 \cdot \text{rand}$
13: $e = \text{temp}$
14: if $(t > 3 \cdot \text{max}_{\text{iter}}/4)$ then
15: $e = 0$
16: end if
17: Evaluate the fitness function for each of the solutions using $f(x_i)$ (Eq. 2) and assign the values for $\text{Food}_\text{pos}$ and $\text{Enemy}_\text{pos}$.
18: for $(i = 1; i < SS; i + +)$ do
19: Find Matrices $\text{Neighbors}_{x_i}$ and $\text{Neighbors}_{\Delta t_{x_i}}$
20: Find the parameters using the following equations
21: $S = \sum_{k=1}^{N} x_i - \text{Neighbors}_{x_i} (k)$
22: $A = \sum_{k=1}^{N} \text{Neighbors}_{\Delta t_{x_i}} (k)$
23: $C_{\text{temp}} = \sum_{k=1}^{N} \text{Neighbors}_{x_i} (k)$
24: $F = \text{Food}_\text{pos} - x_i$
25: $E = \text{Enemy}_\text{pos} + x_i$
26: for $(j = 1; j < D : j + +)$ do
27: $\Delta t_{x(i,j)} = w \Delta t_{x(i,j)} + sS(j) + aA(j) + cC(j) + fF(j) + eE(j)$ \{Update the $\Delta t_{\text{max}}$ parameter of Particles\}
28: if $(\Delta t_{x(i,j)} > \Delta t_{\text{max}}_{\text{max}})$ then
29: $\Delta t_{x(i,j)} = \Delta t_{\text{max}}_{\text{max}}$
30: end if
31: if $(\Delta t_{x(i,j)} < -\Delta t_{\text{max}}_{\text{max}})$ then
32: $\Delta t_{x(i,j)} = -\Delta t_{\text{max}}_{\text{max}}$
33: end if
34: $T = \frac{\Delta t_{x(i,j)}}{\sqrt{1 + (\Delta t_{x(i,j)})^2}}$ \{V3 Transfer function to convert the values to binary\}
35: if $(\text{rand} < T)$ then
36: $x(i,j) = \tilde{x}(i,j)$
37: end if
38: end for
39: end for
40: $t = t + 1$ \{Iteration counter is increasing\}
41: until $(t < \text{max}_{\text{iter}})$ \{Termination criteria are satisfied\}
42: Produce the best solution $\text{Food}_\text{pos}$. 

4.2. Movement of particles. The notations introduced in this section majorly follow [18]. Each of the particles is represented by D dimensional vectors and they are randomly initialized with each individual value being binary.

\[ x_i = (x_{i1}, x_{i2}, \ldots, x_{iD}) \in S \]  

(9)

where S is the available search space.

The velocity is represented by a D dimensional vector and is initialized to zero.

\[ v_i = (v_{i1}, v_{i2}, \ldots, v_{iD}) \]  

(10)

The best personal(local) position recorded by each particle is maintained as

\[ p_i = (p_{i1}, p_{i2}, \ldots, p_{iD}) \in S \]  

(11)

At each iteration each particle changes its position according to its personal best (Pbest) and the global best(gbest) as follows

\[ v_i^{(t+1)} = w v_i^{(t)} + c_1 r_{i1} \cdot (Pbest_i(t) - x_i^{(t)}) + c_2 r_{i2} \cdot (gbest - x_i^{(t)}) \]  

(12)

where \( c_1 \) and \( c_2 \) are acceleration constants and called cognitive and social parameters, respectively. \( r_1 \) and \( r_2 \) are random values \( \in [0, 1] \). \( w \) is called as the inertia weight. It determines how the previous velocity of the particle influences the velocity in the next iteration. The value of \( w \) is determined by the following expression

\[ w = w_{max} - \text{iteration}(\frac{w_{max} - w_{min}}{\text{Max\_iteration}}) \]  

(13)

where \( w_{max} \) and \( w_{min} \) are constants. \( \text{Max\_iteration} \) is the maximum number of iterations to be run.

4.3. The continuous to binary map. The position of each particle is determined by the S shaped transfer function that maps the continuous velocity value to the position of the particle. This is a special sigmoid function that enhances the PSO.

\[ s = \frac{1}{1 + e^{-v_{i,j}}}, i = 1, \ldots, SS, j = 1, \ldots, D \]  

(14)

\[ X(i,j) = \begin{cases} 
1 & \text{if} \ (\text{rand} < s) \\
0 & \text{otherwise} 
\end{cases} \]

4.4. Enhanced particle swarm optimization algorithm. In this section, we present in details the main steps of the binary enhanced particle swarm optimization algorithm as shown in Algorithm 2.

- **Step 1.** Initialize the values of swarm size \( SS(N) \), acceleration constants \( c_1 \) and \( c_2 \), \( w_{max} \), \( w_{min} \), \( v_{max} \) and \( \text{max\_iter} \).
- **Step 2.** The population is randomly initialized as in Eq. 9 and the velocity vectors are initialized to zeros as in Eq. 10.
- **Step 3.** The following steps are repeated until the terminating criteria is met.
  - **Step 1.** Update the value of inertia weight \( w \) according to Eq. 13.
  - **Step 2.** The fitness value of each solution is calculated using \( f(x_i) \).
  - **Step 3.** The personal best solution \( Pbest \) and the global best solution \( gbest \) are assigned.
  - **Step 4.** At each iteration \( t \), the velocity of each particle is calculated according to Eq. 12.
  - **Step 5.** The continuous values are mapped to binary values using the S shaped transfer function mentioned in Eq. 14 and new solutions are generated.
Algorithm 2 Enhanced particle swarm optimization algorithm

1: Set the initial value of swarm size $SS(N)$, acceleration constants $c_1$ and $c_2$, $w_{max}$, $w_{min}$, $v_{max}$ and $max_{iter}$.
2: Randomly initialize the population as $x$ using Eq. 9 for each solution and the velocity vectors $v$ as D dimensional zero vectors as in Eq. 10.
3: Set $t := 0$. {Counter initialization}
4: $w = w_{max} - t \left( \frac{w_{max} - w_{min}}{max_{iter}} \right)$
5: Evaluate the fitness function for each of the solutions using $f(x_i)$(Eq. 2) and assign the values for $Pbest$ and $gbest$.
6: for $(i = 1; i < SS; i + +)$ do
7: $v_i^{(t+1)} = wv_i^{(t)} + c_1r_{i1}(Pbest_{i}(t) - x_i^{(t)}) + c_2r_{i2}(gbest - x_i^{(t)})$
8: end for {Update the velocities of Particles}
9: for $(i = 1; i < SS; i + +)$ do
10: for $(j = 1; j < D; j + +)$ do
11: if $(v(i,j) > v_{max})$ then
12: $v(i,j) = v_{max}$
13: end if
14: if $(v(i,j) < -v_{max})$ then
15: $v(i,j) = -v_{max}$
16: end if
17: $s = \frac{1}{1+e^{-v(i,j)}}$
18: if $(rand < s)$ then
19: $x(i,j) = 1$
20: else
21: $x(i,j) = 0$
22: end if
23: end for
24: end for
25: $t = t + 1$ {Iteration counter is increasing}
26: until$(t < max_{iter})$ {Termination criteria are satisfied}.
27: Produce the best solution $gbest$.

• Step 4. Produce the global best as the best found solution.

5. Hybrid binary dragonfly enhanced particle swarm optimization (HBDESPO) algorithm. We show the main steps of the proposed HBDESPO algorithm for feature selection in Algorithm 3 and summarize in this section.

• Step 1. Split the given data set into three equal sizes of training, validation and testing sets.

• Step 2. Set the initial value of swarm size $SS(N)$, Acceleration constants $c_1$ and $c_2$, $w_{max}$, $w_{min}$, $v_{max}$, $Deltax_{max}$ and $max_{iter}$ and make the dimension $D$ equal to the number of features in the data set.

• Step 3. Randomly initialize the population as $x$ using Eq. 9 for each solution and the $Deltax$ matrix.

• Step 4. The following steps are repeated until the terminating criteria are met.
  – Step 1. Evaluate the fitness function for each of the solutions using the Eq. 2 and assign the values for $Pbest$, $gbest(Food_{pos})$ and $Enemy_{pos}$.
Algorithm 3 Hybrid binary dragonfly enhanced particle swarm optimization algorithm

1: Split the given data set into three equal sizes of training, validation and testing sets.
2: Set the initial value of swarm size $SS(N)$ and make the dimension $D$ equal to the number of features in the data set.
3: Set acceleration constants $c_1$ and $c_2$, $w_{max}$, $w_{min}$, $v_{max}$, $Deltax_{max}$ and $max_{iter}$.
4: Randomly initialize the population as $x$ using Eq. 9 for each solution and the $Deltax$ matrix where each element takes a binary value.
5: Set $t := 0$. \{Counter initialization\}
6: Evaluate the fitness function for each of the solutions using the Eq. 2 and Assign the values for $Pbest$, $gbest$ and $Enemy_pos$ \{The fitness function for feature selection\}
7: $Food_pos = gbest$
8: Run BDA algorithm as given in Algorithm 1
9: Update the values of $Pbest$ and $gbest$.
10: Give the updated $Deltax$ values as initial velocities for EPSO.
11: for $(i = 1; i < SS; i + +)$ do
12: \[ v_{i}(t+1) = w_{i}(t) + c_1 r_1 (Pbest_i(t) - x_{i}(t)) + c_2 r_2 (gbest - x_{i}(t)) \]
13: end for \{Update the velocities of Particles\}
14: for $(i = 1; i < SS, i + +)$ do
15: for $(j = 1; j < D; j + +)$ do
16: if $(v(i,j) > v_{max})$ then
17: \[ v(i,j) = v_{max} \]
18: end if
19: if $(v(i,j) < -v_{max})$ then
20: \[ v(i,j) = -v_{max} \]
21: end if
22: \[ s = \frac{1}{1+e^{-v_{i}(j)}} \]
23: if $(rand < s)$ then
24: \[ x(i,j) = 1 \]
25: else
26: \[ x(i,j) = 0 \]
27: end if
28: end for
29: end for
30: New $Deltax$ matrix = updated velocities.
31: $t = t + 1$. \{Iteration counter is increasing\}
32: until $(t < max_{iter})$. \{Termination criteria are satisfied\}
33: Produce the best solution $gbest$.

- **Step 2.** Run the BDA algorithm.
- **Step 3.** Update the values of $Pbest$ and $gbest$.
- **Step 4.** The updated $Deltax$ matrix corresponds to the initial velocity information in EPSO.
- **Step 5.** Run the EPSO algorithm as mentioned in Algorithm 2 with the new population, $Pbest$ and $gbest$ and updated velocities.
– **Step 6.** Update the $\Delta t$ matrix to the new velocities got by EPSO.

- **Step 5.** Produce the global best as the best found solution.

The combination of the concepts in the binary dragonfly and the binary particle swarm optimization algorithms that are described in the previous sections are combined in this section to derive an algorithm that can benefit from their coexistence. The dragonfly algorithm has the ability to obtain diverse solutions with its formation of static swarms and the enhanced PSO converging to the global best solution creates synergy in the implementation of the hybrid algorithm which results in increased performance. The hybrid algorithm avoids excessive exploitation in the individual algorithms and uses dual exploration technique to reach the solution. In the HBDEPSO algorithm, the decoupling of the velocity vectors of the dragonflies and the particles leads us to an interesting formulation. The velocity vectors are updated independently for both the particles, according to the weighted combination of the personal and global best solutions, and the dragonflies, which are instantaneous in nature. This does not direct one algorithm with the results obtained from the other algorithm but rather allows both of them to explore the search space alternatively and this form of decoupling and alternation is also why the personal and global solutions are only updated once per the whole iteration (after the enhanced particle swarm update) and not updated after the binary dragonfly algorithm.

Some interesting insights are derived from the fact that the velocities are decoupled. The velocities use different ways to work towards the same goal which is beneficial for the hybrid algorithm because it derives from the diversity of the solutions in each iteration which is also the main philosophy behind hybridizing algorithms. It has to be kept in mind that choosing the hyperparameters is very important for getting good solutions and can be accomplished by a simple grid search or a random search over the hyperparameter space.

6. **Feature selection.** The feature selection problem is defined in Section 2. For a feature vector of size $N$, the number of different feature combinations would be $2^N$, which is a huge space to search exhaustively. So the proposed hybrid metaheuristic Algorithm is used to adaptively search the feature space and produce the best feature combination. The fitness function used is the one given in Eq. 2

$$\text{fitness} = \alpha E_R(D) + \beta \frac{|R|}{|C|}$$

(15)

where $E_R(D)$ is the error rate of the classifier, $R$ is the length of the selected feature subset and $C$ is the total number of features. $\alpha \in [0, 1]$ and $\beta = 1 - \alpha$ are constants used to control the weights of classification accuracy and feature reduction.

6.1. **Classifier.** K-nearest neighbor (KNN) [7] is a common simple method used for classification. KNN is a supervised learning algorithm that classifies an unknown sample instance based on the majority vote of its K-nearest neighbors. Here, a wrapper approach to feature selection is used which uses KNN classifier as a guide for the same. Classifiers do not use any model for K-nearest neighbors and are determined solely based on the minimum distance from the current query instance to the neighboring training samples. In this proposed system, the KNN is used as a classifier to ensure robustness to noisy training data and obtain best feature combinations. A single dimension in the search space represents individual feature and hence the position of a particle represents a single feature combination or solution.
| Dataset         | # of Attributes | # of Instances |
|-----------------|-----------------|----------------|
| Zoo             | 16              | 101            |
| WineEW          | 13              | 178            |
| IonosphereEW    | 34              | 351            |
| WaveformEW      | 40              | 5000           |
| BreastEW        | 30              | 569            |
| Breastcancer    | 9               | 699            |
| Congress        | 16              | 435            |
| Exactly         | 13              | 1000           |
| Exactly2        | 13              | 1000           |
| HeartEW         | 13              | 270            |
| KrvskpEW        | 36              | 3196           |
| M-of-n          | 13              | 1000           |
| SonarEW         | 60              | 208            |
| SpectEW         | 60              | 208            |
| Tic-tac-toe     | 9               | 958            |
| Lymphography    | 18              | 148            |
| Dermatology     | 34              | 366            |
| Echocardiogram  | 12              | 132            |
| hepatitis       | 19              | 155            |
| LungCancer      | 56              | 32             |

7. Experimental results. We test the proposed binary HBDESPO algorithm on 20 datasets in Table 1 taken from the UCI machine learning repository [13] and compare it with other algorithms like Binary versions of dragonfly, enhanced particle swarm optimization, GA, bat and grey wolf. Also, we compare the algorithm with HBEPSOND, where the order of implementation of the two algorithms is reversed. We select the datasets to have variety in number of instances and features to test for varied data. We divide the datasets into three sets: training, validation and testing. We select the value of K as 5 based on the trial and error. We use the training set to evaluate the KNN on the validation set through this algorithm to guide the feature selection process. We use only the test data for the final evaluation of the best selected feature combination. We report the global and optimizer-specific parameter setting in Table 2. We set the parameters according to either domain specific knowledge or trial and error. We explain the evaluation criteria in Subsection 7.1

7.1. Evaluation criteria. We divide the datasets into 3 sets of training, validation and testing. We run the algorithm repeatedly $M = 10$ times for statistical significance of the results. The following measures [12] are recorded from the validation data:

1. **Mean fitness function** is the average of the fitness function value obtained from running the algorithm $M$ times. The Mean fitness function is calculated as shown in Eq. 16.

$$Mean = \frac{1}{M} \sum_{i=1}^{M} g_i^*$$  \hspace{1cm} (16)
Table 2. Parameter setting.

| Parameter                        | Value                      |
|----------------------------------|----------------------------|
| No of iterations ($max_{iter}$)  | 70                         |
| No of search agents ($n$)        | 5                          |
| Dimension ($D$)                  | No. of features in the data|
| Search domain                    | [0 1]                      |
| No of runs ($M$)                 | 10                         |
| $w_{max}$                        | 0.9                        |
| $w_{min}$                        | 0.4                        |
| $\Delta t_{max}$                 | 6                          |
| $c_1$                            | 2                          |
| $c_2$                            | 2                          |
| $v_{max}$                        | 6                          |
| $\beta$ in fitness function     | 0.01                       |
| $\alpha$ in fitness function    | 0.99                       |

where $g^*_i$ is the best fitness value obtained at run $i$.

2. **Best fitness function** is the minimum of the fitness function value obtained from running the algorithm $M$ times. The best fitness function is calculated as shown in Eq. 17.

$$\text{Best} = \min_{i=1}^{M} g^*_i$$  \hspace{1cm} (17)

where $g^*_i$ is the best fitness value obtained at run $i$.

3. **Worst fitness function** is the maximum of the fitness function value obtained from running the algorithm $M$ times. The Worst fitness function is calculated as shown in Eq. 18.

$$\text{Worst} = \max_{i=1}^{M} g^*_i$$  \hspace{1cm} (18)

where $g^*_i$ is the best fitness value obtained at run $i$.

4. **Standard deviation** gives the variation of the fitness function value obtained from running the algorithm $M$ times. It is an indicator of the stability and robustness of the algorithm. Larger values of standard deviation would suggest wandering results where as smaller value suggests the algorithm converges to the same value most of the times. The Standard deviation is calculated as shown in Eq. 19.

$$\text{Std} = \sqrt{\frac{1}{M-1} \sum_{i=1}^{M} (g^*_i - \text{Mean})^2}$$  \hspace{1cm} (19)

where $g^*_i$ is the best fitness value obtained at run $i$.

5. **Mean Feature selection ratio** ($FSR$) is the mean of the ratio of the number of selected features to the total number of features when an algorithm is run $M$ times. The Mean Feature selection ratio is calculated as shown in Eq. 20.

$$FSR = \frac{1}{M} \sum_{i=1}^{M} \frac{\text{size}(g^*_i)}{D}$$  \hspace{1cm} (20)
where $g^*_i$ is the best fitness value obtained at run $i$, $size(g^*_i)$ gives the number of features selected and $D$ is the total number of features.

6. **Average F-score** is a measure that evaluates the performance of a chosen feature subset. It requires that in the data spanned by the feature combination the distance between data points in different classes be large and of those in the same class be as small as possible. The Fischer index for a given feature is calculated as in Eq. 21 [21].

$$F_j = \sum_{k=1}^{C} \frac{n_k (\mu^j_k - \mu^j)^2}{(\sigma_j)^2}$$

(21)

$$(\sigma_j)^2 = \sum_{k=1}^{C} n_k (\sigma^j_k)^2$$

(22)

where $F_j$ is the Fischer index for $j$, $\mu^j$ is the mean of the entire data for feature $j$, $(\sigma^j)^2$ is defined as in Eq. 22, $n_k$ is the size of class $k$, $\mu^j_k$ is the mean of class $k$ for feature $j$, $(\sigma^j_k)^2$ is the variance of class $k$ for feature $j$. The average F-score is calculated by taking the average of values obtained from $M$ runs for only the selected features.

7.2. Results. We compare the proposed binary version of the HBDESPO algorithm with the binary dragonfly algorithm, binarized from [19] and the enhanced particle swarm optimization [18] which are the individual algorithms that are used in this paper and other optimizers which have been used for feature selection before such as binary bat [20] and binary grey wolf algorithms [12]. The results are tabulated as follows.

Table 3 outlines the performance of the algorithms using the fitness function mentioned in Eq. 2 in the minimization mode. The table shows the average fitness obtained over $M$ runs and is calculated using Eq. 16. The best performance is achieved by the proposed binary version of the HBDESPO algorithm proving its ability to search the feature space effectively. This indicator tells us that our algorithm is consistently able to attain good fitness values when compared to other algorithms.

Similar results are seen in Table 4 and Table 5 that outline the best and the worst fitness function obtained over $M$ runs and is calculated using Eqs. 17 and 18, respectively. These support the observations from the previous table indicating that our algorithm not only achieves good performance on average but also the best fitness value and the worst fitness value attained are the minimum amongst all the algorithms.

For testing the stability, robustness and the repeatability of convergence of these stochastic algorithms the standard deviation of the fitness values over $M$ runs is recorded as per Eq. 19 in Table 6. The table shows that the HBDESPO algorithm has the ability to converge repeatedly irrespective of the random initialization.

The best selected feature combinations by the algorithms are also allowed to run on the test data and the average classification accuracy and the average feature selection ratio over $M$ runs is recorded using Eqs. 20 and 21, respectively as shown in Tables 7 and 8. As can be seen from these tables, the HBDESPO algorithm is able to select the minimum number of features and yet maintain the classification accuracy. This shows the capability of the HBDESPO algorithm to satisfy both
the objectives of optimization i.e. achieve a high classification accuracy and at the same time chose as fewer number of features for this as possible.

The consolidated performance can be seen in Fig 1 and 2. It can be observed that the proposed algorithm achieves high classification accuracy while keeping feature selection ratio to a minimum. Also, low fitness values are obtained which don’t vary a lot across different runs.
To analyze the separability and closeness of the selected features Fischer score of these features is calculated as shown in Eq. 22. The average over $M$ runs is recorded in Table 9. As shown in the table, HBDESPO algorithm achieves superior data compactness in comparison with the other algorithms.
Table 7. Average performance of the selected features by different algorithms

| Dataset      | HBDESPO | BDA  | EPSO | BGA  | BBA  | BGWO2 | HBEPSOD |
|--------------|---------|------|------|------|------|-------|---------|
| Zoo          | 0.844   | 0.788| 0.791| 0.863| 0.799| 0.851 | 0.852   |
| Wine EW      | 0.916   | 0.923| 0.881| 0.886| 0.726| 0.896 | 0.888   |
| IonosphereEW | 0.835   | 0.799| 0.829| 0.828| 0.817| 0.824 | 0.810   |
| WaveformEW   | 0.823   | 0.807| 0.809| 0.806| 0.779| 0.819 | 0.806   |
| BreastEW     | 0.949   | 0.944| 0.931| 0.892| 0.842| 0.908 | 0.926   |
| Breastcancer | 0.960   | 0.956| 0.956| 0.957| 0.957| 0.957 | 0.958   |
| Congress     | 0.945   | 0.931| 0.943| 0.915| 0.893| 0.928 | 0.935   |
| Exactly      | 0.895   | 0.798| 0.884| 0.687| 0.647| 0.680 | 0.846   |
| Exactly2     | 0.746   | 0.739| 0.738| 0.734| 0.711| 0.732 | 0.736   |
| HeartEW      | 0.815   | 0.81 | 0.776| 0.711| 0.648| 0.702 | 0.811   |
| KruskpEW     | 0.959   | 0.954| 0.958| 0.906| 0.772| 0.917 | 0.958   |
| M-of-n       | 0.978   | 0.949| 0.975| 0.892| 0.719| 0.843 | 0.957   |
| SonarEW      | 0.705   | 0.658| 0.682| 0.694| 0.678| 0.682 | 0.682   |
| SpectEW      | 0.762   | 0.752| 0.757| 0.750| 0.755| 0.777 | 0.747   |
| Tic-tac-toe  | 0.748   | 0.745| 0.740| 0.734| 0.647| 0.713 | 0.737   |
| Lymphography | 0.406   | 0.417| 0.354| 0.416| 0.422| 0.379 | 0.411   |
| Dermatology  | 0.958   | 0.940| 0.952| 0.95  | 0.902| 0.908 | 0.945   |
| Echocardiogram| 0.875   | 0.893| 0.906| 0.852| 0.861| 0.877 | 0.863   |
| Hepatitis    | 0.819   | 0.788| 0.813| 0.798| 0.788| 0.880 | 0.803   |
| LungCancer   | 0.481   | 0.427| 0.390| 0.409| 0.343| 0.345 | 0.345   |
| Average      | 0.821   | 0.801| 0.803| 0.784| 0.730| 0.776 | 0.801   |

Table 8. Average selected feature ratio by different algorithms

| Dataset      | HBDESPO | BDA  | EPSO | BGA  | BBA  | BGWO2 | HBEPSOD |
|--------------|---------|------|------|------|------|-------|---------|
| Zoo          | 0.293   | 0.331| 0.356| 0.412| 0.512| 0.473  | 0.4      |
| Wine EW      | 0.284   | 0.338| 0.4  | 0.315| 0.538| 0.516  | 0.338    |
| IonosphereEW | 0.367   | 0.397| 0.388| 0.402| 0.526| 0.541  | 0.397    |
| WaveformEW   | 0.633   | 0.666| 0.709| 0.676| 0.634| 1      | 0.752    |
| BreastEW     | 0.241   | 0.283| 0.241| 0.290| 0.480| 0.470  | 0.3      |
| Breastcancer | 0.411   | 0.422| 0.511| 0.566| 0.511| 0.644  | 0.544    |
| Congress     | 0.306   | 0.337| 0.325| 0.412| 0.493| 0.575  | 0.318    |
| Exactly      | 0.469   | 0.507| 0.507| 0.561| 0.538| 0.576  | 0.523    |
| Exactly2     | 0.392   | 0.392| 0.492| 0.4  | 0.546| 0.8    | 0.415    |
| HeartEW      | 0.391   | 0.407| 0.407| 0.415| 0.492| 0.430  | 0.4      |
| KruskpEW     | 0.486   | 0.475| 0.502| 0.530| 0.513| 0.633  | 0.516    |
| M-of-n       | 0.515   | 0.530| 0.476| 0.576| 0.446| 0.923  | 0.515    |
| SonarEW      | 0.44    | 0.413| 0.463| 0.42 | 0.521| 0.533  | 0.475    |
| SpectEW      | 0.413   | 0.454| 0.463| 0.425| 0.481| 0.529  | 0.440    |
| Tic-tac-toe  | 0.555   | 0.555| 0.666| 0.511| 0.577| 0.866  | 0.533    |
| Lymphography | 0.39    | 0.438| 0.416| 0.4  | 0.461| 0.535  | 0.45     |
| Dermatology  | 0.5     | 0.411| 0.511| 0.479| 0.494| 0.544  | 0.5      |
| Echocardiogram| 0.225  | 0.233| 0.266| 0.283| 0.508| 0.483  | 0.25     |
| Hepatitis    | 0.273   | 0.273| 0.321| 0.261| 0.515| 0.431  | 0.294    |
| LungCancer   | 0.35    | 0.353| 0.423| 0.380| 0.498| 0.526  | 0.357    |
| Average      | 0.397   | 0.411| 0.442| 0.334| 0.514| 0.601  | 0.436    |

These tables show that the HBDESPO algorithm outperforms the other algorithms with respect to all of the assessment indicators. It can also be seen that it performs much better when compared to its switched version HBEPSOD algorithm. This leads us to believe that the dragonfly algorithm is powerful in exploring the
search space and the enhanced particle swarm optimization algorithm aids in exploiting the reduced feature space. The algorithm also majorly benefits from dual exploration because of the decoupled velocities and outperforms the other algorithms compared with. It is clear that the individual binary algorithms (EPSO and binary dragonfly) don’t perform as well as the hybrid version which is influenced by synergy. The binary bat and the binary grey wolf also lag behind with respect to performance indicators when compared to the proposed algorithm which can be attributed to the novel formulation in this paper.

8. Conclusion. In this paper, we propose a new hybrid binary metaheuristic algorithm with dragonfly algorithm and enhanced particle swarm optimization algorithm in order to solve feature selection problems. We call the proposed algorithm by hybrid binary dragonfly enhanced particle swarm optimization (HBDESPO) algorithm. The two algorithms come together to give better solutions than each of them individually. In order to verify the robustness and the effectiveness of the
proposed algorithm, we apply it on 20 feature selection problems. We perform the evaluation of the proposed algorithm using a set of evaluation criteria to assess different aspects of the proposed system. The experimental results show that the proposed algorithm is a promising algorithm with its ability to search the feature space effectively. The given algorithm was also run on test data and observations.
show higher performance of the selected features when compared to the other optimizers. The Fischer index table reveals better separability. It is also noted from the values of standard deviation that the algorithm has the robustness to repeatedly converge to similar solutions therefore a powerful ability to solve feature selection problems better than other algorithms in most cases.

REFERENCES

[1] D. K. Agrafiotis and W. Cedeno, Feature selection for structure-activity correlation using binary particle swarms, *Journal of Medicinal Chemistry*, 45 (2002), 1098–1107.
[2] H. Banati and M. Bajaj, Fire fly based feature selection approach, *IJCSI International Journal of Computer Science Issues*, 8 (2011).
[3] D. Bell and H. Wang, A formalism for relevance and its application in feature subset selection, *Mach. Learn.*, 41 (2000), 175–195.
[4] B. Xue, M. Zhang, W. Browne and X. Yao, A survey on evolutionary computation approaches to feature selection, *IEEE Transaction on Evolutionary Computation*, 20 (2016), 606–626.
[5] G. Chandrashekar and F. Sahin, A survey on feature selection methods, Electrical and Microelectronic Engineering, Rochester Institute of Technology, Rochester, NY 14623, USA, 2013.
[6] B. Chizi, L. Rokach and O. Maimon, A survey of feature selection techniques, Encyclopedia of Data Warehousing and Mining, seconded, *IGI Global*, (2009), 1888–1895.
[7] L. Y. Chuang, H. W. Chang, C. J. Tu and C. H. Yang, Improved binary PSO for feature selection using gene expression data, *Comput. Biol. Chem.*, 32 (2008), 29–38.
[8] G. Coath and S. K. Halgamuge, A comparison of constraint-handling methods for the application of particle swarm optimization to constrained nonlinear optimization problems, *Proceedings of IEEE Congress on Evolutionary Computation 2003 (CEC 2003)*, Canbella, Australia, (2003), 2419–2425.
[9] C. A. Coello Coello, E. H. Luna and A. H. Aguirre, Use of particle swarm optimization to design combinational logic circuits, *International Conference on Evolvable Systems*, (2003), 398–409.
[10] C. Cotta, A study of hybridisation techniques and their application to the design of evolutionary algorithms, *AI Communications*, 11 (1998), 223–224.
[11] R. C. Eberhart and J. Kennedy, A new optimizer using particle swarm theory. *Proceedings of the Sixth International Symposium on Micromachine and Human Science*, Nagoya, Japan, (1995), 39–43.
[12] E. Emary, H. M. Zawbaa, C. Grosan and A. E. Hassanien, Binary grey wolf optimization approaches for feature selection, *Neurocomputing, Elsevier*, 172 (2016), 371–381.
[13] A. Frank and A. Asuncion, *UCI Machine Learning Repository*, 2010.
[14] J. Huang, Y. Cai and X. Xu, A hybrid genetic algorithm for feature selection wrapper based on mutual information, *Pattern Recognition Letters archive*, 28 (2007), 1825–1844.
[15] J. Kennedy, R. C. Eberhart and Y. Shi, *Swarm Intelligence*, Morgan Kaufmann, SanMateo, CA, 2001.
[16] S. Khalid, A survey of feature selection and feature extraction techniques in machine learning, Science and Information Conference (SAI), 2014.
[17] R. A. Krohling, H. Knidel and Y. Shi, Solving numerical equations of hydraulic problems using particle swarm optimization, *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002)*, Honolulu, Hawaii USA, 2002.
[18] S. Mirjalili and A. Lewis, S-shaped versus V-shaped transfer functions for binary Particle Swarm Optimization, *Swarm and Evolutionary Computation*, 9 (2012), 1–14.
[19] S. Mirjalili, Dragonfly algorithm: A new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems, *Neural Computing and Applications*, 27 (2016), 1053–1073.
[20] R. Y. M. Nakamura, L. A. M. Pereira, K. A. Costa, D. Rodrigues, J. P. Papa and X.-S. Yang, Binary bat algorithm for feature selection, *Conference on Graphics, Patterns and Images*, (2012), 291–297.
[21] Q. Gu, Z. Li and J. Han, *Generalized Fisher Score for Feature Selection*, In Proc. of the 27th Conference on Uncertainty in Artificial Intelligence (UAI), Barcelona, Spain, 2011.
[22] E. G. Talbi, A taxonomy of hybrid metaheuristics, *Journal of Heuristics*, 8 (2002), 541–565.
[23] D. Wolpert and W. Macready, No free lunch theorems for optimization, *IEEE Transactions on Evolutionary Computation*, 1 (1997), 67–72.

Received November 2017; revised January 2018.

*E-mail address: Mtawhid@tru.ca
E-mail address: kdsouza1496@gmail.com*