Feature Selection with a Backtracking Search Optimization Algorithm

Konstantinos Sikelis[1]
[1]*Dept. of Cultural Technology and Communication University of the Aegean*
Mytilene, Greece
cti20004@ct.aegean.gr

George E. Tsekouras[2]
[2]*Dept. of Cultural Technology and Communication University of the Aegean*
Mytilene, Greece
gtsek@aegean.gr

***Abstract*—Feature selection carries significance in the outcome of any classification or regression task. Exercising evolutionary computation algorithms in feature selection has led to the construction of efficient discrete optimization algorithms. In this paper, a modified backtracking search algorithm is employed to perform wrapper-based feature selection, where two modifications of the standard backtracking search algorithm are adopted. The first one concentrates on utilizing a particle ranking operator regarding the current population. The second one focuses on removing the case of using a single particle on the mutation process. Then, the implementation of the above algorithm in feature selection is carried out in terms of two general frameworks, which originally were developed for the particle swarm optimization. The first framework is based on the binary and the second on the set-based particle swarm optimization. The experimental analysis shows that the above variants of the backtracking search algorithm perform equally well on the classification of several datasets.**

***Index Terms*—Backtracking search optimization, feature selection, binary search, set-based search**

I. Introduction

Feature selection (*FS*) is defined as the problem of choosing an optimal subset of features for use in a classification or regression model. It consists of two main components, namely a search technique for proposing new feature subsets and an evaluation function for scoring them.

Evolutionary algorithms have been used extensively to search through the space of possible features. These algorithms are based on three key concepts, namely *particles*, representing the candidate solutions to the problem, *positions*, which are the values of the particles at each iteration and *velocities*, that are directions along which particles' positions are required to change. In the context of FS particles represent different feature combinations.

Particle Swarm Optimization (*PSO*) [1] has been extensively applied to the FS problem, due to its success in continuous optimization. Comprehensive surveys on the use of PSO for solving the FS problem can be found in [2], [3], [4] and [5]. Two well studied classes of discrete PSO algorithms are the binary PSO (*BPSO*) ([6], [7], [8], [9], [10], [11], [12], [13]) and the set-based PSO ([14], [15], [16], [17]). In the BPSO family, particle positions are strings of bits where bit-values of 1 and 0 mark selected and discarded features respectively. Set-based PSO algorithms define particle positions and velocities as sets and construct interactions that lead to new velocity-position pairs.

Recently, Civicioglou *et al,* proposed a new continuous optimization method, called Backtracking Search Optimization (*BSA*) [18]. According to authors the algorithm outperformed several popular evolutionary algorithms including PSO on a wide range of tests. In [19], Tsekouras *et al,* proposed a modified BSA algorithm (*MBSA*) and successfully used it to train a neuro-fuzzy network for modelling shoreline realignment.

In this study, we investigate the feasibility of employing BSA for combinatorial optimization within the context of feature selection. As a starting point, we develop two variations of the novel binary PSO algorithm (*NBPSO*) in [20], and the set-based PSO (*SBPSO*) in [21], where PSO is replaced with MBSA. The new algorithms are compared against their PSO counterparts on feature selection for the classification of datasets from the UCI machine learning repository [22].

The rest of the paper is organized as follows. In section 2, the PSO, BPSO, NBPSO and SBPSO algorithms are briefly summarized. Then BSA and MBSA are reviewed and the proposed binary and set-based versions of discrete MBSA are described. In the fourth section, the performance of the two algorithms is discussed. Conclusions and future work are presented in the last section.

II. Algorithm Review

A. Particle Swarm Optimization

The basic variant of PSO [1] starts with a set of randomly generated particle position-velocity pairs. At each iteration the current position, the best known position for each particle i ($\boldsymbol{P_{pi}}$) and the best known position among all particles ($\boldsymbol{P_g}$), are combined into a new particle velocity-position pair. Algorithm 1 lists the pseudo code for the basic PSO algorithm within the context of minimization. Four parameters, namely ω, ϕ_p, ϕ_g and l_r, are used to tune the performance of PSO. The first three control how the current position and velocity, and the personal and global best positions, affect the new velocity,

Algorithm 1: Particle Swarm Optimization

```
   /* Initialization                                          */
1  P_1 ← U(b_low, b_up);
2  V_1 ← U(b_low − b_up, b_up − b_low) ;
3  P_p1 ← P_1 ;
4  P_g ← P_p1 ;
5  for i = 2 to N_particles do
6  |   P_i ← U(b_low, b_up);
7  |   V_i ← U(b_low − b_up, b_up − b_low) ;
8  |   P_pi ← P_i ;
9  |   if Cost(P_i) < Cost(P_g) then P_g ← P_i ;
10 end for
   /* Loop until certain termination criteria are met  */
11 while Convergence == false do
12 |   for i = 1 to N_particles do
13 |   |   r_p, r_g ← U(0, 1);
14 |   |   V_i ← ωV_i + φ_p r_p ⊙ (P_pi − P_i) + φ_g r_g ⊙ (P_g − P_i);
15 |   |   P_i ← P_i + l_r V_i;
16 |   |   if Cost(P_i) < Cost(P_pi) then
17 |   |   |   P_pi ← P_i;
18 |   |   |   if Cost(P_i) < Cost(P_g) then P_g ← P_i ;
19 |   |   end if
20 |   end for
21 end while
```

while the learning rate l_r controls the effect of the new velocity on the new position.

The first binary PSO algorithm was proposed by Eberhart *et al,* in [23]. The main idea is to retain the continuous nature of the velocity and introduce the sigmoid function as a means to associate continuous velocities with discrete positions. Then for all features j equation 15 in algorithm 1 becomes

$$\boldsymbol{P_i}[j] = (U(0,1) < \frac{1}{1+e^{-\boldsymbol{V_i}[j]}} \ \mathbf{?}\ 1 : 0) \tag{1}$$

where the conditional operator from the C programming language is used to denote the *if-else* statement, according to the following syntax

$$Var = Condition\ \mathbf{?}\ Value\ if\ True : Value\ if\ False$$

Intuitively, in BPSO the velocity can be regarded as the probability that a particular position element is 1 or 0. However, such an interpretation is problematic ([8], [20]). In continuous PSO, velocities direct the current particle position towards the optimum and big absolute values indicate that big change is required, while a zero velocity is indicative of convergence. In BPSO, big velocity values direct, with high probability, position bits to values 0 or 1, thus hindering exploration. On the other hand, a zero velocity causes completely random bit assignments to the new position, which adversely affects exploitation.

A binary PSO algorithm attempting to overcome the drawbacks of the original BPSO was presented in [20]. The proposed algorithm, called Novel Binary PSO (*NBPSO*), combines the probabilistic interpretation of velocity from BPSO with the physical interpretation as rate of change from the continuous PSO. This is achieved by introducing two intermediate probability 'velocity' vectors, $\boldsymbol{V_i^1}$ and $\boldsymbol{V_i^0}$, for each particle i, which express the probability that a bit should be flipped (to 1 and 0 respectively).

At each iteration, both vectors are updated according to the intuitive rule that, for a given bit value b (0 or 1) of the j_{th} bit of a best position (personal or global), the probability of setting the corresponding bit of the new position to b should increase, while the probability of setting it to the complement value $\tilde{b}$ should decrease. Specifically,

$$\begin{aligned} \boldsymbol{V_i^1}[j] &\leftarrow \omega \boldsymbol{V_i^1}[j] + d^1_{ij,p} + d^1_{ij,g} \\ \boldsymbol{V_i^0}[j] &\leftarrow \omega \boldsymbol{V_i^0}[j] + d^0_{ij,p} + d^0_{ij,g} \end{aligned} \tag{2}$$

where

$$\begin{aligned} d^b_{ij,w} &= c_w r_w \\ d^{\tilde{b}}_{ij,w} &= -c_w r_w \end{aligned} \tag{3}$$

In the above, the subscript w takes values p and g and denotes the personal best or the global best position, c_w are positive user selected parameters and r_w random numbers $\in [0,1]$.

Finally, the new value for every bit is calculated by applying the equation:

$$\boldsymbol{P_i}[j] = U(0,1) < \frac{1}{1+e^{-\boldsymbol{V_i^c}[j]}} \ \mathbf{?}\ \tilde{\boldsymbol{P_i}}[j] : \boldsymbol{P_i}[j], \tag{4}$$

where the chosen velocity $\boldsymbol{V_i^c}[j]$ of the jth bit is the one calculated for the complement value to its current value *i.e.,*

$$\boldsymbol{V_i^c}[j] = \boldsymbol{P_i}[j] == 0\ \mathbf{?}\ \boldsymbol{V_i^1}[j] : \boldsymbol{V_i^0}[j] \tag{5}$$

The set-based algorithm in [21] (*SBPSO*) takes a different approach to combinatorial optimization with PSO. Here, the position $\boldsymbol{P_i}$ of particle i is a member of the power set $P(D)$, where D is the domain of discourse, while the velocity $\boldsymbol{V_i}$ is defined as a set of operations, namely element additions and removals, which lead to a new position. Interactions between positions and velocities are directed by the following six operators:

1) *Addition of two Velocities,* $\boldsymbol{V_1} \oplus \boldsymbol{V_2}$, is defined as the union of two operation sets and yields a new velocity, $\boldsymbol{V_{1\oplus 2}} = \boldsymbol{V_1} \cup \boldsymbol{V_2}$
2) *Difference of Two Positions,* $\boldsymbol{P_1} \ominus \boldsymbol{P_2}$, is defined as the set of operations, which convert $\boldsymbol{P_2}$ to $\boldsymbol{P_1}$, *i.e.,* addition of elements of $\boldsymbol{P_1}$ not in $\boldsymbol{P_2}$ and removal of elements of $\boldsymbol{P_2}$ not in $\boldsymbol{P_1}$, and yields a new velocity $\boldsymbol{V_{1\ominus 2}} = (\{+\} \times \boldsymbol{P_1} \setminus \boldsymbol{P_2}) \cup (\{-\} \times \boldsymbol{P_1} \setminus \boldsymbol{P_2})$
3) *Multiplication of a velocity by a scalar,* $\alpha \otimes \boldsymbol{V}$, is defined as a random subset of $\boldsymbol{V}$ with $\lfloor \alpha |\boldsymbol{V}| \rfloor$ elements. It follows, $\alpha \in [0,1],\ 0 \otimes \boldsymbol{V} = \emptyset$ and $1 \otimes \boldsymbol{V} = \boldsymbol{V}$.
4) *Addition of a velocity and a position,* $\boldsymbol{V} \boxplus \boldsymbol{P}$, is defined as the action of applying all operations in the velocity set to the position and yields a new position $\boldsymbol{P}_{\boldsymbol{V}\boxplus\boldsymbol{P}} = \boldsymbol{V}(\boldsymbol{P})$.
5) *Removal of elements in the intersection* $\boldsymbol{S_I} = \boldsymbol{P} \cap \boldsymbol{P_p} \cap \boldsymbol{P_g}$ *from position* $\boldsymbol{P}$, $\beta \odot^- \boldsymbol{S_I}$, is defined as the set of deletions (*i.e.,* a new velocity) of $N_{\beta,\boldsymbol{S_I}}$ elements in $\boldsymbol{S_I}$ from $\boldsymbol{P}$, where

$$N_{\beta,\boldsymbol{S}} = \min(|\boldsymbol{S}|, \lfloor \beta \rfloor + \mathbb{1}) \tag{6}$$

and $\mathbb{1} = U(0,1) < \beta - \lfloor \beta \rfloor \mathbf{?} 1 : 0$.
6) *Addition of elements outside the union* $\boldsymbol{S_U} = \boldsymbol{P} \cup \boldsymbol{P_p} \cup \boldsymbol{P_g}$ *to position* $\boldsymbol{P}$, $\beta \odot_k^+ \overline{\boldsymbol{S}}_{\boldsymbol{U}}$, is defined as the

set of additions of $N_{\beta,\overline{\boldsymbol{S}}_{\boldsymbol{U}}}$ elements in $\overline{\boldsymbol{S}}_{\boldsymbol{U}}$ to $\boldsymbol{P}$, where $N_{\beta,\overline{\boldsymbol{S}}_{\boldsymbol{U}}}$ is given from eq. 6. In order to select each new element, a random element from $\overline{\boldsymbol{S}}_{\boldsymbol{U}}$ is added to the position and the objective function is evaluated. This is repeated k times, and the element which achieved the best score is marked for addition. The whole process is repeated, until all $N_{\beta,\overline{S}_U}$ elements are selected (*k-tournament*).

With these six defined operations the continuous PSO can be readily applied to combinatorial optimization. In particular, the velocity and position update equations (lines 14 and 15 in algorithm 1) become:

$$\begin{aligned}\boldsymbol{V_i} \leftarrow \phi_p r_p \otimes (\boldsymbol{P_{pi}} \ominus \boldsymbol{P_i}) \oplus \phi_g r_g \otimes (\boldsymbol{P_g} \ominus \boldsymbol{P_i}) \\ \oplus (\phi_a r_a \odot_k^+ \overline{\boldsymbol{S}}_{\boldsymbol{Ui}}) \oplus (\phi_r r_r \odot^- \boldsymbol{S_{Ii}})\end{aligned} \tag{7}$$

and

$$\boldsymbol{P_i} \leftarrow \boldsymbol{P_i} \boxplus \boldsymbol{V_i}, \tag{8}$$

where $\phi_p, \phi_g \in [0,1],\ \phi_a, \phi_r \in [0, |U|],\ r_i \in U(0,1)$, $\boldsymbol{S_{Ii}} = \boldsymbol{P_i} \cap \boldsymbol{P_{pi}} \cap \boldsymbol{P_g}$ and $\overline{\boldsymbol{S}}_{\boldsymbol{Ui}} = D \setminus \boldsymbol{P_i} \cup \boldsymbol{P_{pi}} \cup \boldsymbol{P_g}$.

B. Backtracking Search Optimization

Backtracking Search Optimization (BSA) [18] employs a *historic* population, *i.e.,* a set of previous positions, to guide the evolution of an initial random population. At each iteration a previous position, which is randomly assigned to each particle, is used to calculate the particle's velocity vector as its difference from the particle's current position. Then, similarly to PSO, a new candidate position (*Mutant*) is obtained by combining the current position with the calculated velocity. If any mutated elements lie outside the boundaries of allowed values, then they are reset to a random value within this range.

Contrary to PSO, the candidate positions are not directly evaluated. Instead they are further modified by a crossover phase, which aims to increase the diversity of the new trial population. In particular, the mutation of a set of randomly selected elements in the candidate position vector is discarded and those elements remain unchanged. A coin flip decides whether the mutation will be accepted for only one or a larger random number of elements. A second coinflip, at the beginning of each iteration, decides whether to retain the previous *historic* population or to refresh it by replacing it with the current population. BSA's behaviour is affected by two main parameters, namely the learning rate l_r, controlling the effect of the velocity on the new candidate position and the mixrate m_r, which influences the amount of mixing between the mutant and the current position. Algorithm 2 lists the pseudo code of BSA.

In [19], Tsekouras *et al,* argue that the use of the *historic* population and the lack of a strategy to increase the population diversity in the mutation phase, might cause the standard BSA to have an inferior balance between exploitation and exploration, thus exhibiting poor convergence characteristics, such as slow or premature convergence. As a remedy they propose three modifications.

Algorithm 2: Backtracking Search Optimization

```
   /* Initialization                                                    */
1  for i = 1 to N_particles do
2      P_i ← U(b_low, b_up);
3      Ph_i ← U(b_low, b_up);
4  end for
   /* Loop until certain termination criteria are met                  */
5  while Convergence == false do
       /* Coinflip for new historical population                       */
6      if U(0,1) < U(0,1) then Ph ← P;
       /* Obtain new historical population                             */
7      Ph ← Permute(Ph);
       /* Mutation                                                     */
8      for i = 1 to N_particles do
9          ε ← N(0,1);
10         Pm_i ← P_i + l_r ε(Ph_i − P_i);
           /* Boundary Control                                         */
11         for j = 1 to N_features do
12             if Pm_i[j] < b_low || Pm_i[j] > b_up then
13                 Pm_i[j] = b_low + U(0,1)(b_up − b_low)
14             end if
15         end for
16     end for
       /* Crossover                                                    */
17     m_0 ← m_r N_features;
       /* Coinflip to allow single element mutation                    */
18     if U(0,1) < U(0,1) then m_0 ← 0;
19     for i = 1 to N_particles do
20         m_i ← m_0 == 0 ? 1 : m_0 U(0,1));
           /* Random mixing vector with m_i zeros and
              N_features − m_i ones                                     */
21         b ← BooleanVector(m_i);
22         for j = 1 to N_features do
23             Pc_i[j] ← b[j] == 1 ? P_i[j] : Pm_i[j] ;
24         end for
25     end for
26     for i = 1 to N_particles do
27         if Cost(Pc_i) < Cost(P_i) then P_i ← Pc_i ;
28     end for
29 end while
```

The first modification pertains to the mutation operation and allows the utilization of the particle ranking in the current population. In particular, a probability p_{ri} is assigned to each current position $\boldsymbol{P_i}$, according to the formula

$$p_{ri} = 1 - \frac{Rank(\boldsymbol{P_i})}{N_{particles}} \tag{9}$$

A random number, representing a probability threshold, is generated and an individual is randomly picked, out of the subset of particles with probabilities higher than the threshold probability. This step is repeated, until a new *rank* population Pr with $N_{particles}$ is formed. For each particle a velocity vector is calculated as the difference of the current position from the corresponding position in the Pr population. The mutation formula (line 10 in algorithm 2) is modified accordingly to include this second velocity, namely

$$\boldsymbol{Pm_i} \leftarrow \boldsymbol{P_i} + l_r \epsilon(\boldsymbol{Ph_i} - \boldsymbol{P_i}) + \beta(\boldsymbol{Pr_i} - \boldsymbol{P_i}), \tag{10}$$

where $\beta \in U(-1,1)$.

The second proposed modification applies to the crossover operation, where the case of mutating a single element is removed. Specifically, the m_0 assignment in line 18 of algorithm 2 is replaced with

$$m_0 \leftarrow N_{features} \tag{11}$$

Finally, the boundary control of the standard algorithm is also changed. When a mutated position element is out of

bounds, then instead of being randomly placed within the range of allowed values, it is set to the boundary. Line 13 of algorithm 2 becomes

$$\boldsymbol{Pm_i}[j] = \boldsymbol{Pm_i}[j] < b_{low} \ ? \ b_{low} : \\ (\boldsymbol{Pm_i}[j] > b_{up} \ ? \ b_{up} : \boldsymbol{Pm_i}[j]) \quad (12)$$

C. Binary and Set-based Backtracking Search Optimization

The fact that MBSA and PSO share similar update equations (equation 10 and lines 14, 15 in algorithm 1), allows for the construction of binary and set-based variations of NBPSO and SBPSO respectively, by simply replacing the PSO algorithm with MBSA.

Indeed, both updates yield a new position by combining a pair of velocities, defined as the distance of the current position from a previous one. In both algorithms, the previous positions capture good feature combinations, which influence the particle evolution. In the case of PSO, the two velocities utilize the personal and best known positions. In the case of MBSA, the occasional refreshing of the historic population, by replacing it with the current population (line 6 in algorithm 2), implies that the historic population gradually improves. In addition, the rank population contains the best particles from the current population.

Consequently, the incorporation of MBSA in NBPSO and SBPSO is straight forward. Specifically, the PSO personal best and global best positions in equations 2 and 7 are replaced by the MBSA positions from the historic and the rank populations, respectively. Henceforth, we refer to the new variations as NBBSA and SBBSA.

III. Experimental Evaluation

We compare the new algorithms, against their PSO counterparts, on feature selection for a series of classification problems from the UCI repository. The underlying classifier is a simple neural network consisting of two layers, namely a full rank linear layer with leaky ReLU activation, followed by a second full rank linear layer with Sigmoid and SoftMax activation for the cases of binary and multi class classifications, respectively. Accordingly, cross entropy is used as the loss function.

Initially, all datasets are inspected and columns with little or no information (*e.g.,* ids, large number of missing values) are discarded. Afterwards, a reference run is performed, where the network is trained using all features. The performance of the network is evaluated with either a test set, if it is available, or 10-fold cross validation. Tables I, II and III list information on the datasets, the size and network configuration and the training setup, which were used for the reference run and gave the best reference performance.

For feature selection, the accuracy of the classification is used as the objective function of the optimization. In all cases, the evolving population consists of 16 individuals and the number of epochs was set to 30 (except for Bands with 100 epochs and Autism with 10). All parameters were set to 1.0 and no tuning was performed. Additionally, for the set-based version of discrete PSO and BSA algorithms, k-tournament was not performed as it was found to increase excessively the runtime of the feature selection process. Instead, $N_{\beta,\overline{S}_U}$ elements out of $\overline{S}_U$ were randomly picked. All runs were carried out on an Intel(R) Core(TM) i9-9960X CPU @ 3.10GHz with 16 threads. The code was written in C and parallel processing was implemented using the OpenMP API [24].

Table I
Dataset Attributes

Dataset	Nrecords		#Features		#Classes
	Train	Test	Total	Discarded	
Arrhytmia	452		280	5	13
Audiology	200	26	70	1	24
Bands	540		39		2
Autism	704		20		2
BreastTissue	106		9		6
Dermatology	366		34		6
Glass	214		10	1	6
Hill-Valley	606	606	100		2
Horse-Colic	300	68	36	2	2
Ionosphere	351		34		2
Musk1	476		168	2	2
Myocardical	1700		124	2	8
Adult	32561	16281	14		2

Table II
Neural Network configuration

Dataset	Layer 1		Layer2	Loss
	In	Out	Activation	Function
Arrhytmia	281	64	LogSoftMax	LogCrossEntropy
Audiology	161	64	LogSoftMax	LogCrossEntropy
Bands	39	16	Sigmoid	BinrayCrossEntropy
Autism	20	16	Sigmoid	BinrayCrossEntropy
BreastTissue	9	9	SoftMax	CrossEntroy
Dermatology	130	16	SoftMax	CrossEntroy
Glass	9	4	SoftMax	CrossEntroy
Hill-Valley	100	16	Sigmoid	BinaryCrossEntroy
Horse-Colic	25	16	Sigmoid	BinaryCrossEntroy
Ionosphere	34	16	Sigmoid	BinaryCrossEntroy
Musk1	166	64	Sigmoid	BinaryCrossEntroy
Myocardical	388	64	SoftMax	CrossEntropy
Adult	14	8	Sigmoid	BinaryCrossEntroy

Table V lists the results of the experiments. All four algorithms were successful in both reducing the number of features and improving the classification accuracy with respect to the reference run, even for the datasets with small number of features and/or high reference accuracy. They all performed comparably in terms of feature reduction, accuracy improvement and total runtime. Set-based algorithms appear to have a slight edge as they managed to achieve a bit higher feature reductions, with similar accuracy to their binary counterparts. Similarly, PSO variants tend to select less features, while BSA versions tend to have a bit higher accuracy.

IV. Conclusion

In this paper, two algorithms were presented, extending the BSA continuous optimization algorithm to combinatorial optimization, and were applied to the feature selection problem. Specifically, the similarity in the update equation between PSO

Table III
TRAINING SETUP

Dataset	Optimizer	Encoding	Scaling	Validation
Arrhytmia	Grad. Desc.	One-Hot	No	10-fold
Audiology	Grad. Desc.	One-Hot	No	Test Set
Bands	ADAM	Label	MinMax	10-fold
Autism	ADAM	Label	MinMax	10-fold
BreastTissue	ADAM	One-Hot	MinMax	10-fold
Dermatology	ADAM	One-Hot	MinMax	10-fold
Glass	ADAM	One-Hot	No	10-fold
Hill-Valley	Grad. Desc.	Label	MinMax	Test Set
Horse-Colic	ADAM	Label	MinMax	Test Set
Ionosphere	ADAM	Label	MinMax	10-fold
Musk1	Grad. Desc.	Label	MinMax	10-fold
Myocardical	Grad. Desc.	One-Hot	MinMax	10-fold
Adult	Grad. Desc.	Label	MinMax	Test Set

Table IV
REFERENCE RUN

Dataset	Ncycles	Learning Rate	Batch Size	Accuracy (Mean %)
Arrhytmia	1000	0.005	64	75.5
Audiology	5000	0.005	64	84.0
Bands	1000	0.001	64	52.7
Autism	5000	0.01	64	98.4
BreastTissue	50000	0.001	16	79.8
Dermatology	2000	0.01	64	95.1
Glass	15000	0.01	64	61.4
Hill-Valley	5000	0.1	32	64.5
Horse-Colic	12000	0.001	32	80.9
Ionosphere	25000	0.01	64	96.0
Musk1	3000	0.01	64	91.4
Myocardical	300	0.01	64	91.3
Adult	2500	0.01	32	84.6

and MBSA allowed for the straight forward incorporation of MBSA into two PSO based discrete optimization algorithms, namely NBPSO and SBPSO, by, effectively, replacing PSO with MBSA.

Performance of the two new BSA based variations was tested against their PSO counterparts on feature selection for the classification of several datasets from the UCI repository. Although, our analysis is by no means exhaustive, test results indicate that BSA can be a viable alternative to PSO for feature selection and discrete optimization in general.

In the future, we plan to dig deeper into the performance of the presented algorithms both by tuning their internal parameters and trying them with more datasets, classifiers, and problems in general (*e.g.,* regression). Furthermore, the popularity of PSO in discrete optimization offers a large number of algorithms were BSA can be integrated.

REFERENCES

[1] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, 1995, pp. 39–43.

[2] V. Kothari, J. Anuradha, S. Shah, and P. Mittal, "A survey on particle swarm optimization in feature selection," in *Global Trends in Information Systems and Software Applications*, P. V. Krishna, M. R. Babu, and E. Ariwa, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 192–201.

[3] N. Omar, F. Jusoh, R. Binti, and M. Othman, "Review of feature selection for solving classification problems," *Journal of Research and Innovation in Information Systems*, pp. 64–70, 01 2013.

[4] C. Yun, B. Oh, J. Yang, and J. Nang, "Feature subset selection based on bio-inspired algorithms," *J. Inf. Sci. Eng.*, vol. 27, pp. 1667–1686, 09 2011.

[5] B. X., M. Zhang, and W. N. Browne, "New fitness functions in binary particle swarm optimisation for feature selection," in *2012 IEEE Congress on Evolutionary Computation*, 2012, pp. 1–8.

[6] Y. Shi and R. Eberhart, "Fuzzy adaptive particle swarm optimization," in *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*, vol. 1, 2001, pp. 101–106 vol. 1.

[7] L. Wang, X. Wang, J. Fu, and L. Zhen, "A novel probability binary particle swarm optimization algorithm and its application," 2008.

[8] H. Nezamabadi-pour, M. R. Shahrbabaki, and M. M. Farsangi, "Binary particle swarm optimization: Challenges and new solutions," *The CSI Journal on Computer Science and Engineering*, vol. 6, no. 1, 2008.

[9] L. Chuang, S. Tsai, and C. Yang, "Improved binary particle swarm optimization using catfish effect for feature selection," *Expert Systems with Applications*, vol. 38, no. 10, pp. 12 699–12 707, 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417411005732

[10] A. El-Maleh, A. T. Sheikh, and S. M. Sait, "Binary particle swarm optimization (bpso) based state assignment for area minimization of sequential circuits," *Applied Soft Computing*, vol. 13, no. 12, pp. 4832–4840, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494613002688

[11] L. Kumar and K. Bharti, *An Improved BPSO Algorithm for Feature Selection*. Springer, Singapore, 01 2019, pp. 505–513.

[12] D. Liu, Z. Xiao, H. Li, X. Hu, and M. O.P., "Accurate parameter estimation of a hydro-turbine regulation system using adaptive fuzzy particle swarm optimization," *Energies*, vol. 12, no. 20, p. 3903, oct 2019.

[13] B. H. Nguyen, B. Xue, P. Andreae, and M. Zhang, "A new binary particle swarm optimization approach: Momentum and dynamic balance between exploration and exploitation," *IEEE Transactions on Cybernetics*, vol. 51, no. 2, pp. 589–603, 2021.

[14] C. B. Veenhuis, "A set-based particle swarm optimization method," in *Parallel Problem Solving from Nature – PPSN X*, G. Rudolph, T. Jansen, N. Beume, S. Lucas, and C. Poloni, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 971–980.

[15] M. Neethling and A. Engelbrecht, "Determining rna secondary structure using set-based particle swarm optimization," in *2006 IEEE Congress on Evolutionary Computation, CEC 2006*. IEEE, 01 2006, pp. 1670 – 1677.

[16] W. Chen, J. Zhang, H. S. H. Chung, W. Zhong, W. Wu, and Y. Shi, "A novel set-based particle swarm optimization method for discrete optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 2, pp. 278–300, 2010.

[17] T. Hino, S. Ito, T. Liu, and M. Maeda, "Set-based particle swarm optimization with status memory for knapsack problem," *Artificial Life and Robotics*, vol. 21, pp. 98–105, 2015.

[18] P. Çivicioglu, "Backtracking search optimization algorithm for numerical optimization problems," *Appl. Math. Comput.*, vol. 219, pp. 8121–8144, 2013.

[19] A. Chatzipavlis, G. Tsekouras, V. Trygonis, A. Velegrakis, J. Tsimikas, A. Rigos, T. Hasiotis, and C. Salmas, "Modeling beach realignment using a neuro-fuzzy network optimized by a novel backtracking search algorithm," *Neural Computing and Applications*, vol. 31, pp. 1747–1763, 2018.

[20] M. A. Khanesar, M. Teshnehlab, and M. A. Shoorehdeli, "A novel binary particle swarm optimization," in *2007 Mediterranean Conference on Control Automation*, 2007, pp. 1–6.

[21] A. P. Engelbrecht, G. J., and J. Langeveld, "Set based particle swarm optimization for the feature selection problem," *Engineering Applications of Artificial Intelligence*, vol. 85, pp. 324–336, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197619301472

[22] "Uci machine learning repository," https://archive.ics.uci.edu/.

[23] J. Kennedy and R. Eberhart, "A discrete binary version of the particle swarm algorithm," in *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, vol. 5, 1997, pp. 4104–4108 vol.5.

[24] "The openmp api specification for parallel programming," https://www.openmp.org/.

Table V
FEATURE NUMBER AND ACCURACY AFTER FEATURE SELECTION

Dataset	**Algorithm**	**#Features Excluded**	**% Feature Reduction**	**Accuracy (Mean %)**	**%Accuracy Improvement**	**Elapsed (m:s)**
Arrhytmia	SBMBSA	98	35.6	76.9	1.3	24:00
	SBPSO	**169**	61.4	77.6	2.0	24:00
	NBMBSA	144	52.3	**77.8**	2.2	22:00
	NBPSO	130	47.2	77.3	1.8	22:00
Audiology	SBMBSA	26	37.7	88.5	4.5	06:00
	SBPSO	**27**	39.1	**92.3**	8.3	06:00
	NBMBSA	25	36.2	**92.3**	8.3	05:21
	NBPSO	42	60.1	88.5	4.5	05:15
Bands	SBMBSA	20	51.3	**65.3**	12.6	18:18
	SBPSO	**22**	56.4	64.8	12.0	18:06
	NBMBSA	19	48.7	65.0	12.2	18:30
	NBPSO	19	48.7	63.9	11.1	18:06
Autism	SBMBSA	14	70.0	100	1.6	05:48
	SBPSO	14	70.0	100	1.6	05:15
	NBMBSA	14	70.0	100	1.6	05:24
	NBPSO	14	70.0	100	1.6	05:18
BreastTissue	SBMBSA	4	44.4	80.8	1.0	30:30
	SBPSO	4	44.4	80.8	1.0	32:15
	NBPSO	4	44.4	80.8	1.0	29:00
	NBMBSA	4	44.4	80.8	1.0	30:30
Dermatology	SBMBSA	**20**	58.8	99.7	4.5	06:38
	SBPSO	16	47.0	99.7	4.5	06:23
	NBMBSA	13	38.2	**100**	4.8	06:26
	NBPSO	17	50.0	99.3	4.2	06:14
Glass	SBMBSA	**5**	55.5	70.0	8.6	15:31
	SBPSO	**5**	55.5	70.0	8.6	18:24
	NBMBSA	**5**	55.5	70.0	8.6	15:30
	NBPSO	4	44.4	70.0	8.6	15:54
Hill-Valley	SBMBSA	**52**	52.0	77.2	12.7	02:27
	SBPSO	43	43.0	77.4	12.9	02:23
	NBMBSA	49	49.0	77.2	12.7	02:20
	NBPSO	46	46.0	**78.5**	14.0	02:17
Horse-Colic	SBMBSA	9	26.5	**85.2**	4.4	02:08
	SBPSO	**11**	32.3	**85.2**	4.4	02:03
	NBMBSA	8	23.5	**85.2**	4.4	02:06
	NBPSO	11	32.3	83.8	2.9	02:02
Ionosphere	SBMBSA	13	38.2	99.4	3.4	45:03
	SBPSO	**20**	58.8	**99.7**	3.7	43:28
	NBMBSA	13	38.2	99.1	3.1	44:16
	NBPSO	13	38.2	**99.7**	3.7	43:11
Musk1	SBMBSA	77	46.4	**96.2**	4.9	37:35
	SBPSO	47	28.3	94.6	3.2	35:35
	NBMBSA	79	47.6	94.4	3.0	33:12
	NBPSO	**81**	48.8	95.1	3.7	32:46
Myocardical	SBMBSA	49	40.1	92.1	0.8	29:24
	SBPSO	46	37.7	**92.2**	0.9	30:31
	NBMBSA	49	40.1	91.9	0.6	27:32
	NBPSO	**54**	44.2	92.1	0.8	27:57
Adult	SBMBSA	4	28.6	85.1	0.5	35:18
	SBPSO	4	28.6	85.1	0.5	34:16
	NBMBSA	**6**	42.9	85.1	0.5	34:54
	NBPSO	**6**	42.9	85.1	0.5	34:02