Feature Selection Using Binary Particle Swarm Optimization with Time Varying Inertia Weight Strategies

Majdi Mafarja
Department of Computer Science, Birzeit University
Birzeit
mmafarja@birzeit.edu

Radi Jarrar
Department of Computer Science, Birzeit University
Birzeit
rjarrar@birzeit.edu

Sobhi Ahmad
Department of Computer Science, Birzeit University
Birzeit
sahmed@birzeit.edu

Ahmed A. Abusnaina
Department of Computer Science, Birzeit University
Birzeit
aabusnaina@birzeit.edu

ABSTRACT

In this paper, a feature selection approach that based on Binary Particle Swarm Optimization (PSO) with time varying inertia weight strategies is proposed. Feature Selection is an important preprocessing technique that aims to enhance the learning algorithm (e.g., classification) by improving its performance or reducing the processing time or both of them. Searching for the best feature set is a challenging problem in feature selection process, metaheuristics algorithms have proved a good performance in finding the (near) optimal solution for this problem. PSO algorithm is considered a primary Swarm Intelligence technique that showed a good performance in solving different optimization problems. A key component that highly affect the performance of PSO is the updating strategy of the inertia weight that controls the balance between exploration and exploitation. This paper studies the effect of different time varying inertia weight updating strategies on the performance of BPSO in tackling feature selection problem. To assess the performance of the proposed approach, 18 standard UCI datasets were used. The proposed approach is compared with well regarded metaheuristics based feature selection approaches, and the results proved the superiority of the proposed approach.

CCS CONCEPTS

• **Computing methodologies → Feature selection**;

KEYWORDS

Binary Particle Swarm Optimization, PSO, Feature Selection, Classification, Inertia Weight, Optimization

ACM Reference Format:
Majdi Mafarja, Radi Jarrar, Sobhi Ahmad, and Ahmed A. Abusnaina. 2018. Feature Selection Using Binary Particle Swarm Optimization with Time Varying Inertia Weight Strategies. In *Proceedings of* . ACM, New York, NY, USA, 9 pages.

1 INTRODUCTION

The performance of the classification algorithms is highly sensitive to data dimensionality [29]. Having a large number of features in a dataset increases the chance to mislead the learning algorithm (e.g., classifier) and thus decrease its performance. Feature Selection (FS) is a crucial preprocessing step that can contribute to improving the performance of classification algorithms by eliminating redundant, irrelevant and noisy features [12]. Besides enhancing the performance of classification algorithms, FS contributes to reducing the required computational time.

When designing an FS algorithm, two main issues must be taken into consideration; the first is how to search for the best feature subset, while the second is how to define the goodness of a subset [31]. Considering the goodness of a feature subset, there are two main models that have been widely used in FS literature; filter and wrapper models. In filters, the evaluation process depends mainly on the relations between the features and the dependencies between those features and the class. Some examples of the filter approach are F-score, Information Gain (IG) and Principal Component Analysis (PCA) [26]. In wrapper approaches, the evaluation process depends mainly on the performance of a learning algorithm (e.g., classification) that should be involved in the evaluation process. Since a learning algorithm is involved when evaluating each feature subset, then the optimization process of the wrapper approaches would be slower than filter approaches, however, the wrappers are more appropriate when the performance of a specific learning algorithm is the target.

The second aspect of FS methods is how to search for the best feature subset. Three main strategies can be used; exhaustive (complete), random and heuristic search strategies [31]. On one hand, complete search strategy tends to find all possible feature subsets, evaluate them, and then select the best one. This is obviously a complicated and time-consuming strategy. Formally speaking, if we have a dataset with N features, then 2^N subsets should be generated and evaluated. So, when dealing with large size datasets, this strategy becomes impractical. On the other hand, random search is another strategy to search the features subsets, but, in the worst case, it may try as many solutions as the complete search does without finding the target solution. The third possible strategy that may be used with FS problem is the heuristic search.

This strategy is known to be moderate between the complete and the random search strategies. It starts from a randomly generated solution, then a heuristic value guides the search process towards the target subset. The main aim of heuristic algorithms is to find the (near) optimal solution in a reasonable time [22].

, ,
2018. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00

Many taxonomies were used to classify Metaheuristics (MH) algorithms, one popular taxonomy is following the nature of inspiration [49]. Two main categories are well studied in the literature; Evolutionary Algorithms (EA), Swarm Intelligence (SI). In all EAs, an initial population is randomly generated at the beginning of the optimization process, then this population is updated in an evolutionary manner until satisfying a stopping criterion. The most popular examples of EAs are Genetic Algorithm (GA) [24] and Differential Evolution (DE) [47].

SI algorithms are nature inspired algorithms that mimic the social and self-organization behaviors of the creatures that usually live in groups like fish, birds, particles, etc. [15]. SI algorithms are similar to EAs since they are population-based MH algorithms, where an initial population should be initiated at the beginning of the optimization process, then a certain behavior of the creature that the algorithm was originally inspired from is simulated to update the population. For example, Particle Swarm Optimization (PSO) [14], is a primary SI algorithm that inspires the swarming behavior of the folks of birds. In PSO, each solution in the population represents a particle in the swarm, and the swarming behaviors of the birds were simulated to update the population until satisfying a stopping condition. Ant Colony Optimization (ACO) [13], is another example of SI algorithms that simulate the food foraging behavior of ants in nature. Moreover, the behavior of bees in identifying the food sources and collecting their food in nature was the main inspiration of the Artificial Bees Colony (ABC) algorithm [25].

Recently, many nature-inspired algorithms were proposed to solve various optimization problems. Mussels Wandering Optimization (MWO) algorithm [7] is inspired ecologically by mussels movement behavior seeking nutrition. Flower Pollination Algorithm (FPA) is another algorithm that simulates flower pollination behavior in nature [55]. The MWO and FPA algorithms and their enhanced variants have been applied successfully in the supervised training of both neural networks generations; Artificial Neural Networks [1, 2, 4, 45] and Spiking Neural Networks [3, 5]. Grey Wolf Optimizer (GWO) [44] is a recent SI algorithm that mimics the hierarchical organization of the grey wolves in nature. GWO has been widely used in FS methods with much success. A GWO based FS approach was proposed by [17]. In this approach, an evolutionary operator (i.e., crossover) was incorporated into the GWO process. Whale Optimization Algorithm (WOA), is another SI algorithm that was recently proposed by Mirjalili and Lewis [43]. WOA was successfully applied in many FS methods. Mafarja and Mirjalili proposed two FS methods that employed improved WOA algorithm, the first approach proposed a hybridized the WOA with Simulated Annealing (SA) algorithm [41], while the second proposed an enhanced WOA algorithm by employing some evolutionary operators like crossover and mutation [39]. Another FS method that based on Antlion Optimizer (ALO) was proposed in [38]. More recntly, a FS that adopted Salp Swarm Algorithm (SSA) as a selection mechanism was proposed in [19]. For more filter and wrapper feature approaches, readers can refer to [6, 32–37, 40]

PSO is a primary SI algorithm that has been widely used to solve many optimization problems with much success [42]. Since PSO was originally proposed to solve the continuous optimization problems, a binary version of PSO was proposed in [27] to tackle the binary optimization problem called BPSO. BPSO was used as a search strategy in many FS approaches. Xue et al. [52] proposed FS approaches that based on BPSO with different initialization and updating strategies. A recent FS that is based on an enhanced BPSO with a crossover operator was proposed in [11]. In [9], a wrapper FS approach that is based on BPSO as a search strategy and C4.5 classifier as an evaluator was proposed. In all the previously mentioned approaches, PSO proved its superior performance in comparison to similar FS approaches that use SI and EAs based algorithms.

In any population-based metaheuristic algorithm (e.g., PSO, GWO, ALO), there are two main phases that the optimization process should pass in; exploration (diversification) and exploitation (intensification) [49]. In exploration, the algorithm tries to explore the whole search space with the aim of finding the promising regions that may contain the global optima. However, in exploitation, the algorithm tries to search the neighborhood of each solution found in the exploration phase. Thus, it's important to have more exploration than exploitation at the early stages of the optimization process, while the exploitation becomes more important at the last stages to increase the probability of discovering better solutions, close to those found in the previous phase. Having a good balance between exploration and exploitation has a high impact on the performance of the algorithm.

In PSO, there is only one parameter, called inertia weight (w), that controls the balance between exploration and exploitation. Using a large value for w facilitates exploration, while a small value facilitates exploitation. Thus, a proper adjustment for w is a key issue that affects the performance of BPSO algorithm. In literature, many studies were performed to assess the influence of the updating strategies for such parameters [48]. Harrison et al. [23] empirically investigated different updating strategies for the inertia weight parameter.

In this paper, a wrapper FS approach was proposed, where five different updating strategies (i.e., linear, nonlinear coefficient, decreasing, oscillating and algorithmic) were employed to control the inertia weight parameter in BPSO. In the proposed approach, k-NN classifier was used as an evaluator, the classification accuracy and the number of selected features were incorporated in the fitness function. The proposed approaches were benchmarked using 12 well-known UCI datasets [30]. The experimental results showed a varying performance of BPSO according to the employed updating strategy of w. The produced results proved the effect of using different updating strategies with BPSO and the algorithm's performance. The non-linear updating strategy with coefficient based approach showed the best performance in terms of classification accuracy and the fitness value. Followed by the linear-based approach and which ranked in the first place in terms of the number of selected features. This finding proves that the updating strategies that gradually decrease w are able to balance between the exploration and exploitation mechanisms in BPSO.

The remaining sections of this paper are organized as follows: In Section 2, an overview of PSO algorithm is presented, followed by a description of the proposed approach in Section 3. Section 4 presents the experimental results. Finally, a conclusion and a future direction are provided in Section 5.

2 PARTICLE SWARM OPTIMIZATION

Particle Swarm Optimization is a population-based metaheuristic algorithm that is motivated by the simulation of social behavior such as of the flock of birds and school of fish [46]. PSO is a quite simple algorithm and a new solution from a previous one can be obtained using only two rules [8]. Although it is simple, PSO was found to converge to the optimum in a wide range of optimization problems and is computationally feasible approach [8, 10]. The fundamental concept of PSO is that knowledge is optimized not only by the personal experience but from the social interaction in the population [51].

In PSO, every single particle (i.e., candidate solution) is a point in the d-dimensional search space. Particles have their own memories that record their best experience in the search space along with the best experience encountered by the swarm as a whole to find the best solution [50]. Each individual solution in PSO moves in the search space with a dynamically adjusted velocity that is affected by its own experience and the global experience of other particles.

The initial population of the particles in the swarm is distributed randomly over the search space. The position of each particle is denoted as a vector where D is the dimensionality of the search space. As each particle moves $x_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$ in the search space to find the optimal solution, the velocity of the search $v_i = (v_{i1}, v_{i2}, ..., v_{iD})$. During the movement, particles update their positions and velocity according to their experience and that of their neighbors. Each particle has a memory to record the position where it encountered its best experience and is denoted as *pbest*. The global best, which is the best encounter among all particles, is denoted as *gbest*. The position and velocity are updated for each particle. Both *pbest* and *gbest* help in updating the position and velocity of each particle according to the following equations

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \tag{1}$$

$$v_{id}(t+1) = w * v_{id}(t) + c_1 * r_1 * (p_{id} - x_{id}(t)) + c_2 * r_2 * (p_{gd} - x_{id}(t)) \tag{2}$$

where t represents the t^{th} iteration in the process of evolution. $d \in D$ is the d^{th} dimension in the search space. Variable w is the inertia weight that controls the impact of the previous velocities on the current velocity. The variables c_1 and c_2 are the acceleration (i.e., learning) constants. $r_1, r_2 \in [0, 1]$ are uniformly distributed random numbers. Variable p_{id} is the local best (i.e., *pbest*) and variable p_{gd} is the global best (i.e., *gbest*) in the d^{th} dimension.

The velocity is constrained by a predefined maximum velocity (i.e., v_{max}) and the new velocity $v_{id}^{t+1} \in [-v_{max}, v_{max}]$. The algorithm stops when a predefined stopping criterion is met. The stopping criterion might be a good fitness value or a predefined maximum number of iteration [51].

2.1 Binary Particle Swarm Optimization

PSO was originally applied to solve problems in continuous-numbers search space. However, feature selection, as in many other optimization problems, occur in discrete search spaces. Kennedy and Eberhart [27] developed a binary version of the particle swarm algorithm (BPSO) to tackle optimization problems in discrete domains. In BPSO, the velocity is still updated in the same fashion as in the standard PSO. However, variables $x_{id}, p_{id}, andpgd$ can only have the values 0 or 1. In doing so, velocity would indicate the probability of a particle in the position vector to take the value 1 [53].

In BPSO, the position of the current particle is updated as in Eq. (3) based on the probability value $T(V_t)$ obtained from Eq. (4).

$$x(t+1) = \begin{cases} 1 & \text{If } rand < S(v(t+1)) \\ 0 & \text{Otherwise} \end{cases} \tag{3}$$

where $S(v(t))$ is the *Sigmoid function* as depicted in Fig. 1

$$S(v(t)) = \frac{1}{1 + e^{-v(t)}} \tag{4}$$

where *rand* is a random number $\in [0, 1]$.

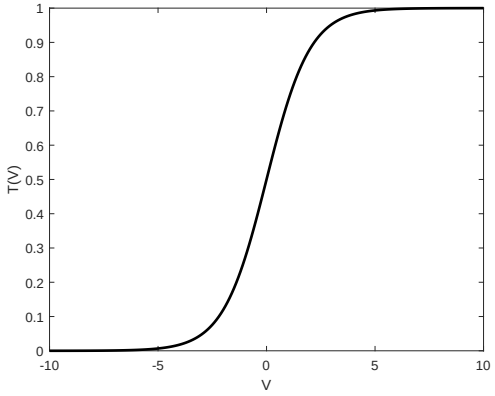


Figure 1: Sigmoid Transfer function

2.2 Binary Particle Swarm Optimization for Feature Selection

In this paper, a wrapper FS 'that employs BPSO algorithm as a search strategy and k-NN classifier as an evaluator is proposed. To formulate the FS problem as an optimization problem, two issues must be considered. The first issue is the solution representation. Since FS is a binary optimization problem, then we choose to represent the solution with a binary vector, where the 1 values indicate that the corresponding feature is selected, otherwise it is not selected. The solution size is the number of features in each dataset. The second issue is designing the fitness function since we adopted the wrapper approach, then in addition to the number of selected features, the classification accuracy must be taken into consideration. Thus, the proposed fitness function is considered both number of selected features and the classification accuracy (see Eq. 5). Since the classification accuracy is to be minimized, the classifier error rate, which is the complement of the classification accuracy, is used in the fitness function.

$$Fitness = \alpha \gamma_R(D) + \beta \frac{|R|}{|C|} \tag{5}$$

where $\gamma_R(D)$ shows the classifiers error rate, |R| represents the number of selected features the feature subset, |C| represents dimensionality of the dataset, and $\alpha \in [1, 0]$, $\beta = (1 - \alpha)$ are two variables to address the importance of each objective (i.e., error rate and selection ration). The values of α, and β were adopted from literature [6, 17].

3 THE PROPOSED APPROACH

In every population-based MH algorithm, the optimization process consists of two primary stages; exploration and exploitation. Having a good balance between those two stages enables the algorithm to converge towards the global optima in a reasonable time. The ideal situation for any algorithm is to employ the exploration operator more than the exploitation operator at the beginning of the search process. This enables the algorithm to explore many regions in the search space, then those regions need to be searched carefully hoping to find the global optima. Usually, in each algorithm, there is one or more parameter(s) that plays this role. Fortunately, PSO has only one parameter that controls this balance; called inertia weight (w).

To achieve this goal, the value of w should be gradually decreased over the course of time (i.e., iteration number) instead of using a fixed value. Different updating mechanisms with different behavior were used in literature. In this paper, five different update strategies that are based on different mechanisms were tested and evaluated as follows:

- *Inertia weight with linearly decreasing strategy*
  Shi and Eberhart [46] presented this method based on the assumption that exploration is favored at early stages of the optimization process, while exploitation is favored at the end. w at each iteration is defined as in Eq.(6).

$$w(t) = w_{max} - (w_{max} - w_{min})\frac{t}{T} \tag{6}$$

  In this strategy, w is decreased linearly from w_{max} at the early iterations to w_{min} at the later. Fig (2a) shows the behavior of this strategy.
- *Inertia weight with non-linear coefficient decreasing strategy*
  The non-linear coefficient decreasing strategy was proposed by Yang et al. [54] to improve the performance of PSO that uses linear updating strategy, w at each iteration is defined as in Eq.(7).

$$w(t) = w_{max} - (w_{max} - w_{min})\left(\frac{t}{T}\right)^{\alpha} \tag{7}$$

  where α is suggested by the authors to be $\alpha = \frac{1}{\pi^2}$. Fig (2b) shows the behavior of this strategy.
- *Inertia weight with decreasing strategy*
  Fan and Chiu [18] proposed this strategy to update the inertia weight as a nonlinear fashion over the iterations as in Eq.(8).

$$w(t) = \left(\frac{2}{t}\right)^{0.3} \tag{8}$$

  Fig (2c) shows the behavior of this strategy.
- *Inertia weight with oscillating strategy*
  The oscillating decreasing strategy employs a sinusoidal function to update w during the search process instead of monotonically decreasing it [28]. The inertia weight (w) at each iteration is defined as in Eq.(9).

$$\text{w(t)=}\begin{cases} w_t = \frac{w_{min}+w_{max}}{2} + \frac{w_{max}-w_{min}}{2} cos\left(\frac{2\pi t(4k+6)}{T}\right) & if\, t < \frac{3T}{4} \\ w_{min} & otherwise \end{cases} \tag{9}$$

  where k is set by the authors to 7. Fig (2d) shows the behavior of this strategy.
- *Inertia weight with logarithmic strategy*
  Gao et al. [21] presented a logarithmic decreasing strategy to update the w value in the course of iterations according to Eq. (10).

$$w(t) = w_{max} + (w_{min} - w_{max}) \times \log_{10}\left(a + \frac{10t}{T}\right) \tag{10}$$

  where a is a constant that set to 1 by the authors. Fig (2e) shows the behavior of this strategy.

4 EXPERIMENTAL SETUP AND RESULTS

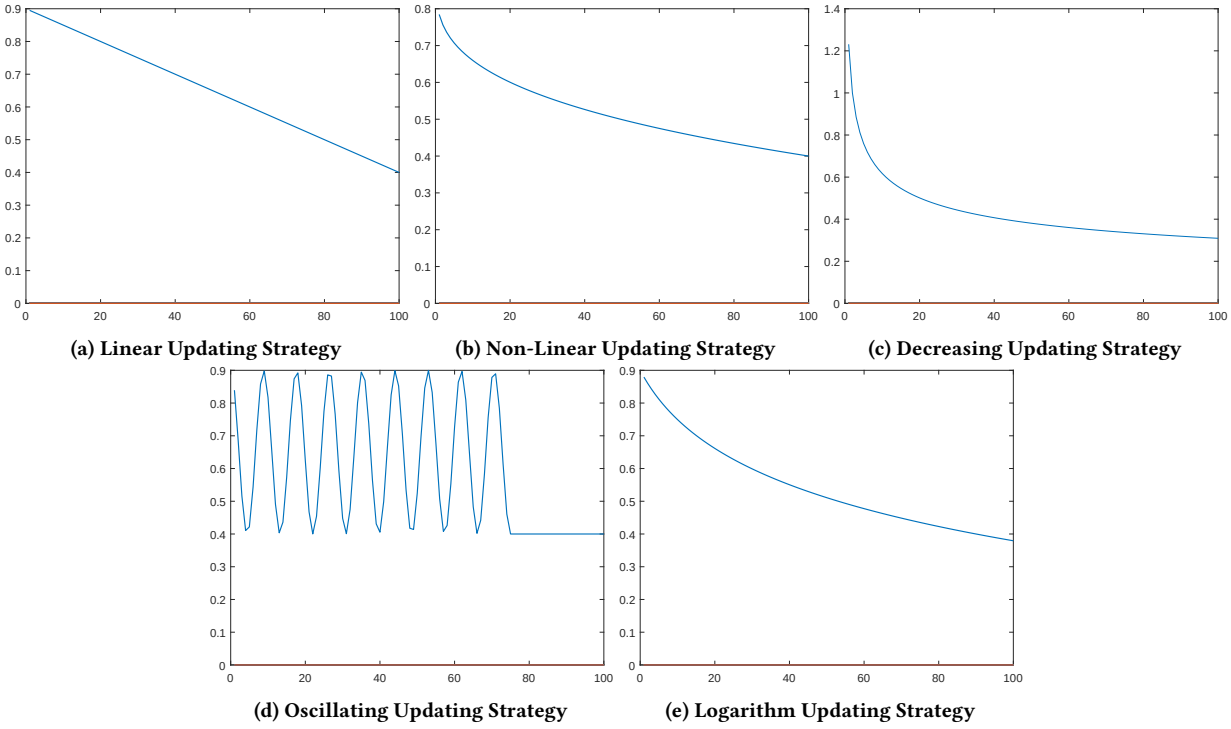
In this section, the results obtained from the proposed approaches were presented and discussed. To assess the performance the proposed approaches we used a set of the well-known datasets from UCI repository [30]. The details of the used datasets are presented in Table 1. The parameters that have been used in all experiments are listed in Table 2. All parameter values have been set either based on some preliminary results or based on previous researches. The proposed approaches were implemented using Matlab, and all experiments were executed on a PC with Intel Core i5 processor, 2.2 GHz CPU and 4 GB of RAM. A KNN (with $K = 5$ [39]) classifier was used to measure the fitness of each feature subset. All datasets were divided into two parts; 80% of the instances were devoted to training and 20% of the instances were used to test the approaches [20]. The presented approaches were evaluated based on several criteria; average classification accuracy (see Eq. 11), average selection size (see Eq. 13), average fitness values (see Eq. 12), and finally the average running time (see Eq. 14).

$$AvgAccuracy = \frac{1}{M}\sum_{j=1}^{M}\frac{1}{N}\sum_{i=1}^{N}(C_i = L_i) \tag{11}$$

where M is the number of runs for an algorithm to find the optimal subset of features, N is the number of dataset instances, C_i is the predictive class, and L_i is the actual class in the labeled data.

$$AvgFitness = \frac{1}{M}\sum_{j=1}^{M} Fit_*^i \tag{12}$$

where M is the number of runs, Fit_*^i is the fitness value of the best solution in the i^{th} run.



(a) Linear Updating Strategy (b) Non-Linear Updating Strategy (c) Decreasing Updating Strategy

(d) Oscillating Updating Strategy (e) Logarithm Updating Strategy

Figure 2: Different Time Varying Updating Strategies for Inertia Weight Parameter in BPSO

$$AvgSize = \frac{1}{M}\sum_{i=1}^{M}\frac{d_i^*}{D} \tag{13}$$

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

$$AvgTime = \frac{1}{M}\sum_{i=1}^{M} RT_i \tag{14}$$

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

Table 1: The used UCI benchmark datasets

Dataset	Number of Features	Number of Samples
Breastcancer	9	699
BreastEW	30	569
Exactly2	13	1000
Lymphography	18	148
M-of-n	13	1000
PenglungEW	325	73
SonarEW	60	208
SpectEW	22	267
KrvskpEW	36	3196
Tic-tac-toe	9	958
Vote	16	300
WaveformEW	40	5000

Table 2: The parameter settings

Parameter	Value
Population size	10
Number of iteration	100
Dimension	Number of features
Number of runs for each technique	30
α in fitness function	0.99
β in fitness function	0.01
a in GWO	[2 0]
w_{min}	0
w_{max}	2

To study the influence of w using different updating strategies on the performance of BPSO algorithm, a comparison between the five updating strategies is provided in this section and in addition to the results of two state-of-the-art FS methods (*i.e.,* BGWO [17] and BALO [16]). Note that the best results are highlighted in boldface in the following tables.

Inspecting the classification accuracy results in Table 3, it can be seen that, in general, the BPSO based approaches obtained the best results compared with BGOW and BALO based approaches. Both BGWO and BALO obtained the best result in one out of 12 datasets while BPSO based approaches obtained the best results on the remaining datasets. Considering BPSO based together, it can be seen that using BPSO with non-linear with coefficient updating

mechanism outperformed other approaches in 42% of the datasets, followed by the linear and the logarithmic based BPSO approaches that obtained the best results in three datasets. Note that those three approaches employ a decreasing inertia weight that depends on the ratio of the current iteration (t) to the maximum number of iteration (T). Decreasing (which is a variant of the non-linear updating mechanism) based approach comes in the third place. This approach decreases the inertia weight during the search process, but the maximum number of iteration is not involved in the updating process. From Fig. (2c), it can be noticed that w is decreased faster than other approaches, which means more exploitation than exploration is performed during the search process while the other approaches have a better balance between the exploration and exploitation. The oscillation based approach comes in the last place in terms of optimizing the classification accuracy. This can be justified due to the behavior of this function where the optimization process oscillates between exploration and exploitation phases in the early iterations. The exploitation phase is emphasized in the last quartile of the iterations.

Table 3: Average classification accuracy for all approaches

Dataset	Linear	Non-linear coefficient	Decreasing	Oscillating	Logarithm	bGWO	bALO
Breastcancer	0.957	**0.984**	0.958	0.964	0.957	0.960	0.943
BreastEW	0.932	0.918	0.942	0.928	0.940	**0.944**	0.936
Exactly2	0.700	0.702	0.653	0.704	**0.735**	0.682	0.727
Lymphography	0.740	0.832	0.808	0.798	**0.854**	0.812	0.774
M-of-n	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.950	0.998
penglungEW	0.931	0.962	0.964	0.772	0.967	0.841	**0.993**
SonarEW	**0.948**	0.900	0.874	0.933	0.911	0.890	0.887
SpectEW	0.717	0.746	**0.879**	0.803	0.745	0.780	0.814
KrvskpEW	0.970	**0.976**	0.969	0.970	0.966	0.945	0.974
Tic-tac-toe	**0.823**	0.702	0.807	0.740	0.759	0.799	0.806
Vote	0.981	**0.984**	0.928	0.803	0.949	0.955	0.936
WaveformEW	0.745	**0.755**	0.742	0.747	0.746	0.735	0.730

Besides the good performance of the linear and non-linear based approaches on the classification accuracy, they recorded the best results in selecting the minimal number of features as can be seen in Table 4. The linear based approach was ranked the first among where it outperformed other approaches on 60 % of the datasets, followed by the non-linear based approach. The decreasing and logarithmic based approaches come in the third place by obtaining the best results in 2 datasets only. The obtained results prove the ability of the linear and non-linear based approaches to select the most informative features that reveal the highest classification accuracy.

Table 4: Average selection size for all approaches

Dataset	Linear	Non-linear coefficient	Decreasing	Oscillating	Logarithm	bGWO	bALO
Breastcancer	5.00	5.40	6.90	4.00	6.00	5.00	**3.53**
BreastEW	**10.73**	13.80	13.60	13.90	12.47	18.00	20.93
Exactly2	6.77	5.20	**4.50**	6.97	6.97	7.50	9.03
Lymphography	**6.73**	7.20	6.80	7.87	7.33	10.60	12.47
M-of-n	**6.00**	**6.00**	**6.00**	**6.00**	**6.00**	9.03	6.97
penglungEW	**114.77**	120.40	136.80	133.77	129.30	154.57	160.17
SonarEW	**24.27**	25.97	24.90	24.53	26.03	36.17	43.53
SpectEW	**6.37**	8.27	7.50	8.43	7.17	12.03	12.87
KrvskpEW	20.63	**18.33**	21.20	20.83	21.50	29.30	27.57
Tic-tac-toe	6.00	**5.00**	9.00	9.00	6.17	8.20	6.20
Vote	**3.17**	3.53	4.10	3.50	4.30	6.87	8.37
WaveformEW	22.13	22.43	21.40	21.67	**20.93**	34.20	35.03

Table 5 shows the average fitness values for all approaches. It can be clearly seen that the best fitness values were obtained by the BPSO based approaches in comparison to others. Moreover, the linear-based approach outperformed other approaches. Next to it is the non-linear and decreasing approaches which come in the second place. Note that BGWO and BALO did not outperform the BPSO approaches in any dataset.

Table 5: Best fitness value for all approaches

Dataset	Linear	Non-linear coefficient	Decreasing	Oscillating	Logarithm	bGWO	bALO
Breastcancer	0.020	**0.008**	0.023	0.012	0.022	0.029	0.026
BreastEW	0.024	0.021	**0.019**	0.020	0.026	0.032	0.035
Exactly2	0.233	0.229	**0.188**	0.209	0.198	0.247	0.241
Lymphography	0.139	0.048	0.062	0.042	**0.024**	0.137	0.139
M-of-n	**0.005**	**0.005**	**0.005**	**0.005**	**0.005**	0.061	0.009
penglungEW	**0.004**	**0.004**	**0.004**	0.189	**0.004**	0.132	0.005
SonarEW	**0.004**	0.044	0.060	0.021	0.018	0.074	0.061
SpectEW	**0.076**	0.082	0.077	0.101	0.168	0.159	0.105
KrvskpEW	0.029	**0.022**	0.023	0.025	0.026	0.047	0.024
Tic-tac-toe	0.173	0.182	**0.145**	0.171	0.193	0.169	0.180
Vote	**0.002**	**0.002**	0.003	**0.002**	0.017	0.021	0.037
WaveformEW	0.210	**0.201**	0.214	0.204	0.203	0.221	0.219

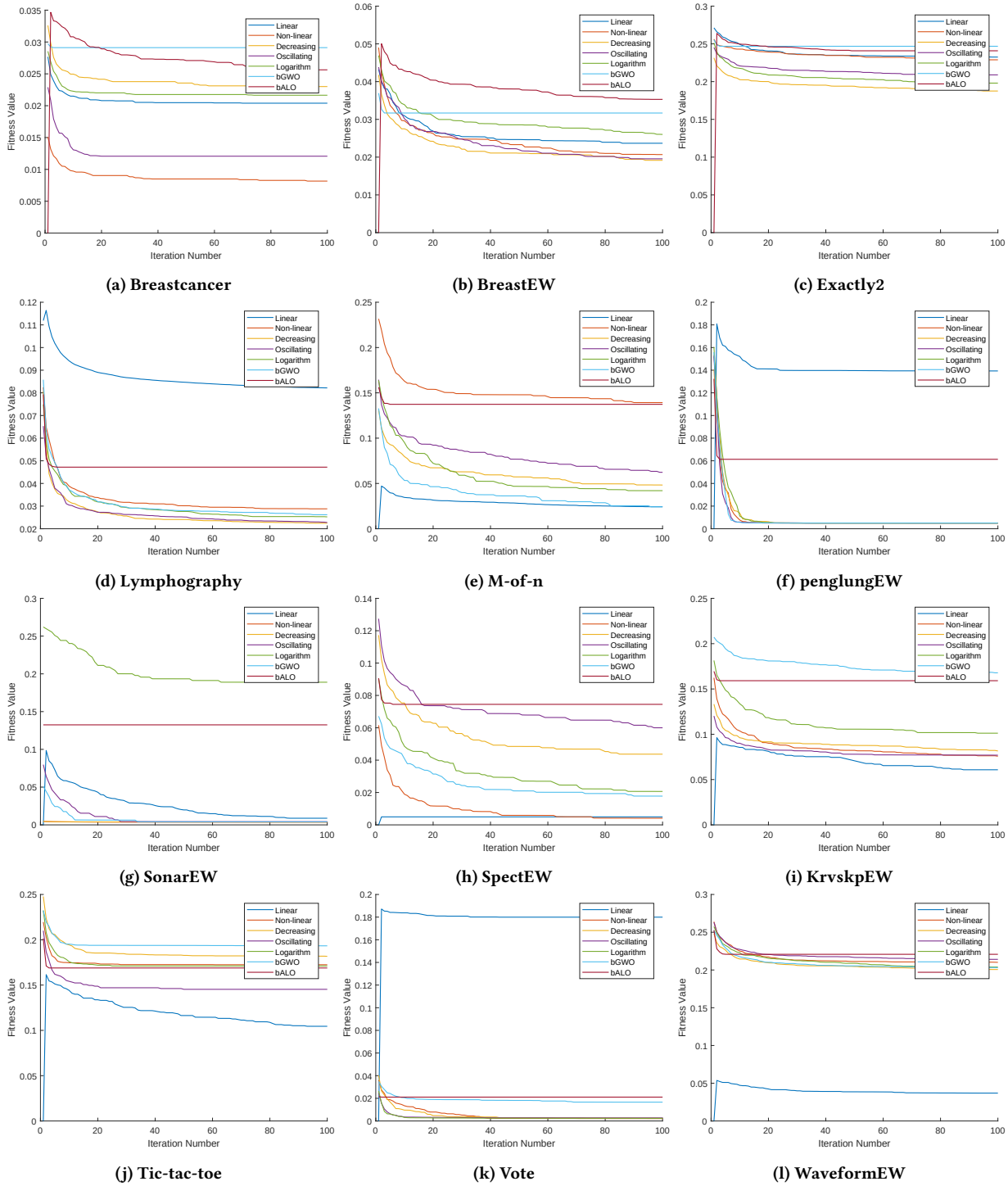
The run-time of the algorithms is another important factor to be considered especially when dealing with the large datasets. In Table 6, it is clear that BPSO methods require lower average runtime to obtain the best results in comparison to the other methods. The comparison of convergence rates is also provided in Fig. 3.

From Fig. 3, it can be observed that the non-linear based approach has a superior convergence rate besides obtaining the best results in terms of fitness and accuracy on 50% of the datasets. Moreover, it has a competitive performance on the remaining datasets. In other words, it takes BPSO fewer iterations to converge and find the (near) optimal solution in comparison to BGWO and BALO.

Table 6: Average running time for all approaches

Dataset	Linear	Non-linear coefficient	Decreasing	Oscillating	Logarithm	bGWO	bALO
Breastcancer	**22.88**	22.99	37.62	38.63	48.69	39.58	43.22
BreastEW	20.87	**20.13**	31.20	32.22	44.25	37.49	42.99
Exactly2	**28.69**	34.36	36.38	35.00	54.25	40.20	59.85
Lymphography	33.09	32.56	25.12	**22.76**	39.22	24.92	35.29
M-of-n	51.28	43.02	33.91	**32.67**	53.49	36.87	55.95
penglungEW	37.46	41.67	23.93	**22.57**	41.53	53.12	37.89
SonarEW	33.00	39.89	22.69	**21.21**	36.32	35.12	37.53
SpectEW	35.80	41.14	25.10	**20.30**	31.74	38.64	34.34
KrvskpEW	278.34	249.07	**164.17**	225.93	237.40	417.86	350.02
Tic-tac-toe	48.42	51.89	**30.54**	53.46	53.20	65.94	41.47
Vote	37.30	36.49	**20.35**	39.69	38.23	38.44	25.00
WaveformEW	681.91	595.56	**580.10**	615.04	640.08	941.50	793.24

Taking the results in this section together, it showed that the updating strategy of the inertia weight parameter (w) influences the robustness and the convergence of BPSO algorithm. In the proposed approaches, five updating strategies were employed to update the inertia weight parameter of the BPSO algorithm. Four strategies depend on decreasing the inertia weight gradually in the course of iterations to allow more exploration than exploitation in the early iterations, while the exploitation is emphasized in the later iterations. The fifth strategy converts between exploration and exploitation in the first 0.75 % of the iterations and the exploitation are employed in the last 0.25 % of the iterations. The



(a) Breastcancer (b) BreastEW (c) Exactly2

(d) Lymphography (e) M-of-n (f) penglungEW

(g) SonarEW (h) SpectEW (i) KrvskpEW

(j) Tic-tac-toe (k) Vote (l) WaveformEW

Figure 3: Convergence curves for BPSO with Different Updating Strategies for the Inertia Weight Parameters.

results showed that the updating strategies that used to decrease the value of w gradually performed better than the oscillating one. Moreover, the non-linear updating strategy was able to outperform other approaches in terms of classification accuracy and fitness values, while it obtained competitive results in terms of the number of selected features. Here, we can remark that the enhanced performance of the proposed approach is because the non-linear update strategy enhanced the ability of BPSO algorithm to balance between exploration and exploitation. Also, the results of this section showed the superior performance of the presented BPSO approaches when compared to the state-of-the-art approaches.

5 CONCLUSIONS

In this paper, a wrapper FS approach using BPSO algorithm was presented. In PSO, there is only one parameter (called inertia weight w) that controls the balance between exploration and exploitation. Having a good balance between these two phases is a key issue in enhancing the performance of any population-based metaheuristic algorithm. In this paper, the influence of using five different updating strategies for w was investigated. For evaluation purposes, 12 UCI benchmark datasets were used. The reported results showed that BPSO methods have shown superior results in comparison to other similar methods by means of average classification accuracy, average selection size, the best fitness value, and the average running time. The extensive analysis of the reported results revealed that the updating strategies that gradually decrease the inertia weight parameter in linear and non-linear fashions could improve the exploration and exploitation behaviors of BPSO for FS tasks.

REFERENCES

[1] Ahmed A Abusnaina and Rosni Abdullah. 2013. Mussels wandering optimization algorithmn based training of artificial neural networks for pattern classification. Proceedings of the 4th International Conference on Computing and Informatics, ICOCI, 78–85.

[2] Ahmed A Abusnaina, Rosni Abdullah, and Ali Kattan. 2014. Enhanced MWO training algorithm to improve classification accuracy of artificial neural networks. In *Recent advances on soft computing and data mining*. Springer, 183–194.

[3] Ahmed A. Abusnaina, Rosni Abdullah, and Ali Kattan. 2015. The Application of Mussels Wandering Optimization Algorithm for Spiking Neural Networks Training. In *1st International Engineering Conference (IEC2014) On Developments in Civil and Computer Engineering Applications*. 197–204.

[4] Ahmed A Abusnaina, Rosni Abdullah, and Ali Kattan. 2018. Self-Adaptive Mussels Wandering Optimization Algorithm with Application for Artificial Neural Network Training. *Journal of Intelligent Systems* (2018).

[5] Ahmed A Abusnaina, Rosni Abdullah, and Ali Kattan. 2018. Supervised Training of Spiking Neural Network by Adapting the E-MWO Algorithm for Pattern Classification. *Neural Processing Letters* (2018), 1–22.

[6] Sobhi Ahmed, Majdi Mafarja, Hossam Faris, and Ibrahim Aljarah. 2018. Feature selection using salp swarm algorithm with chaos. (2018).

[7] Jing An, Qi Kang, Lei Wang, and Qidi Wu. 2013. Mussels wandering optimization: an ecologically inspired algorithm for global optimization. *Cognitive Computation* 5, 2 (2013), 188–199.

[8] Julio Barrera and Carlos A Coello Coello. 2009. A review of particle swarm optimization methods used for multimodal optimization. In *Innovations in swarm intelligence*. Springer, 9–37.

[9] Lucija Brezočnik. 2017. Feature selection for classification using particle swarm optimization. In *Smart Technologies, IEEE EUROCON 2017-17th International Conference on*. IEEE, 966–971.

[10] Girish Chandrashekar and Ferat Sahin. 2014. A survey on feature selection methods. *Computers & Electrical Engineering* 40, 1 (2014), 16–28.

[11] Yonggang Chen, Lixiang Li, Jinghua Xiao, Yixian Yang, Jun Liang, and Tao Li. 2018. Particle swarm optimizer with crossover operation. *Engineering Applications of Artificial Intelligence* 70 (2018), 159–169.

[12] M. Dash and H. Liu. 1997. Feature selection for classification. *Intelligent data analysis* 1, 3 (1997), 131–156.

[13] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. 2006. Ant colony optimization. *IEEE computational intelligence magazine* 1, 4 (2006), 28–39.

[14] Russell Eberhart and James Kennedy. 1995. A new optimizer using particle swarm theory. In *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on*. IEEE, 39–43.

[15] Russell C Eberhart, Yuhui Shi, and James Kennedy. 2001. *Swarm intelligence*. Elsevier.

[16] Eid Emary, Hossam M Zawbaa, and Aboul Ella Hassanien. 2016. Binary ant lion approaches for feature selection. *Neurocomputing* 213 (2016), 54–65.

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

[18] Shu-Kai S Fan and Yi-Yin Chiu. 2007. A decreasing inertia weight particle swarm optimizer. *Engineering Optimization* 39, 2 (2007), 203–228.

[19] Hossam Faris, Majdi M. Mafarja, Ali Asghar Heidari, Ibrahim Aljarah, AlaâĂŹ M. Al-Zoubi, Seyedali Mirjalili, and Hamido Fujita. 2018. An efficient binary Salp Swarm Algorithm with crossover scheme for feature selection problems. *Knowledge-Based Systems* 154 (2018), 43 – 67. https://doi.org/10.1016/j.knosys.2018.05.009

[20] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2001. *The elements of statistical learning*. Vol. 1. Springer series in statistics New York.

[21] Yue-lin Gao, Xiao-hui An, and Jun-min Liu. 2008. A particle swarm optimization algorithm with logarithm decreasing inertia weight and chaos mutation. In *Computational Intelligence and Security, 2008. CIS'08. International Conference on*, Vol. 1. IEEE, 61–65.

[22] Fred W Glover and Gary A Kochenberger. 2006. *Handbook of metaheuristics*. Vol. 57. Springer Science & Business Media.

[23] Kyle Robert Harrison, Andries P Engelbrecht, and Beatrice M Ombuki-Berman. 2016. Inertia weight control strategies for particle swarm optimization. *Swarm Intelligence* 10, 4 (2016), 267–305.

[24] John H. Holland. 1992. *Adaptation in natural and artificial systems*. MIT Press. 211 pages.

[25] Dervis Karaboga and Bahriye Basturk. 2007. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of global optimization* 39, 3 (2007), 459–471.

[26] Alexandr Katrutsa and Vadim Strijov. 2017. Comprehensive study of feature selection methods to solve multicollinearity problem according to evaluation criteria. *Expert Systems with Applications* 76 (2017), 1–11.

[27] James Kennedy and Russell C Eberhart. 1997. A discrete binary version of the particle swarm algorithm. In *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*, Vol. 5. IEEE, 4104–4108.

[28] Kyriakos Kentzoglanakis and Matthew Poole. 2009. Particle swarm optimization with an oscillating inertia weight. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*. ACM, 1749–1750.

[29] Ohbyung Kwon and Jae Mun Sim. 2013. Effects of data set features on the performances of classification algorithms. *Expert Systems with Applications* 40, 5 (2013), 1847–1857.

[30] M. Lichman. 2013. UCI Machine Learning Repository. (2013). http://archive.ics.uci.edu/ml

[31] Huan Liu and Hiroshi Motoda. 2012. *Feature selection for knowledge discovery and data mining*. Vol. 454. Springer Science & Business Media.

[32] Majdi Mafarja and Salwani Abdullah. 2011. Modified great deluge for attribute reduction in rough set theory. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2011 Eighth International Conference on*, Vol. 3. IEEE, 1464–1469.

[33] Majdi Mafarja and Salwani Abdullah. 2013. Investigating memetic algorithm in solving rough set attribute reduction. *International Journal of Computer Applications in Technology* 48, 3 (2013), 195–202.

[34] Majdi Mafarja and Salwani Abdullah. 2013. Record-to-record travel algorithm for attribute reduction in rough set theory. *J Theor Appl Inf Technol* 49, 2 (2013), 507–513.

[35] Majdi Mafarja, Salwani Abdullah, and Najmeh S Jaddi. 2015. Fuzzy population-based meta-heuristic approaches for attribute reduction in rough set theory. *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering* 9, 12 (2015), 2462–2470.

[36] Majdi Mafarja, Ibrahim Aljarah, Ali Asghar Heidari, Abdelaziz I. Hammouri, Hossam Faris, AlaâĂŹM. Al-Zoubi, and Seyedali Mirjalili. 2018. Evolutionary Population Dynamics and Grasshopper Optimization approaches for feature selection problems. *Knowledge-Based Systems* 145 (2018), 25 – 45. https://doi.org/10.1016/j.knosys.2017.12.037

[37] Majdi Mafarja and Derar Eleyan. 2013. Ant colony optimization based feature selection in rough set theory. (2013).

[38] Majdi Mafarja, Derar Eleyan, Salwani Abdullah, and Seyedali Mirjalili. 2017. S-shaped vs. V-shaped transfer functions for ant lion optimization algorithm in feature selection problem. In *Proceedings of the International Conference on Future Networks and Distributed Systems*. ACM, 14.

[39] Majdi Mafarja and Seyedali Mirjalili. 2018. Whale optimization approaches for wrapper feature selection. *Applied Soft Computing* 62 (2018), 441–453.
[40] Majdi M Mafarja, Derar Eleyan, Iyad Jaber, Abdelaziz Hammouri, and Seyedali Mirjalili. 2017. Binary dragonfly algorithm for feature selection. In *New Trends in Computing Sciences (ICTCS), 2017 International Conference on*. IEEE, 12–17.
[41] Majdi M Mafarja and Seyedali Mirjalili. 2017. Hybrid Whale Optimization Algorithm with simulated annealing for feature selection. *Neurocomputing* 260 (2017), 302–312.
[42] Seyedali Mirjalili and Andrew Lewis. 2013. S-shaped versus V-shaped transfer functions for binary particle swarm optimization. *Swarm and Evolutionary Computation* 9 (2013), 1–14.
[43] Seyedali Mirjalili and Andrew Lewis. 2016. The whale optimization algorithm. *Advances in Engineering Software* 95 (2016), 51–67.
[44] Seyedali Mirjalili, Seyed Mohammad Mirjalili, and Andrew Lewis. 2014. Grey wolf optimizer. *Advances in engineering software* 69 (2014), 46–61.
[45] Moh'd Khaled Yousef Shambour, Ahmed A Abusnaina, and Ahmed I Alsalibi. 2018. Modified Global Flower Pollination Algorithm and its Application for Optimization Problems. *Interdisciplinary Sciences: Computational Life Sciences* (2018), 1–12.
[46] Yuhui Shi and Russell C Eberhart. 1999. Empirical study of particle swarm optimization. In *Evolutionary computation, 1999. CEC 99. Proceedings of the 1999 congress on*, Vol. 3. IEEE, 1945–1950.
[47] Rainer Storn and Kenneth Price. 1997. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization* 11, 4 (1997), 341–359.
[48] Mojtaba Taherkhani and Reza Safabakhsh. 2016. A novel stability-based adaptive inertia weight for particle swarm optimization. *Applied Soft Computing* 38 (2016), 281–295.
[49] El-Ghazali Talbi. 2009. *Metaheuristics: from design to implementation*. Vol. 74. John Wiley & Sons.
[50] Gerhard Venter and Jaroslaw Sobieszczanski-Sobieski. 2003. Particle swarm optimization. *AIAA journal* 41, 8 (2003), 1583–1589.
[51] Bing Xue, Mengjie Zhang, and Will N Browne. 2013. Particle swarm optimization for feature selection in classification: A multi-objective approach. *IEEE transactions on cybernetics* 43, 6 (2013), 1656–1671.
[52] Bing Xue, Mengjie Zhang, and Will N Browne. 2014. Particle swarm optimisation for feature selection in classification: Novel initialisation and updating mechanisms. *Applied Soft Computing* 18 (2014), 261–276.
[53] Bing Xue, Mengjie Zhang, Will N Browne, and Xin Yao. 2016. A survey on evolutionary computation approaches to feature selection. *IEEE Transactions on Evolutionary Computation* 20, 4 (2016), 606–626.
[54] Chengwei Yang, Wei Gao, Nengguang Liu, and Chongmin Song. 2015. Low-discrepancy sequence initialized particle swarm optimization algorithm with high-order nonlinear time-varying inertia weight. *Applied Soft Computing* 29 (2015), 386–394.
[55] Xin-She Yang. 2012. Flower pollination algorithm for global optimization. In *International conference on unconventional computing and natural computation*. Springer, 240–249.