Structures 57 (2023) 105280

Contents lists available at ScienceDirect

Structures

journal homepage: www.elsevier.com/locate/structures

Structural health monitoring via a group-theoretic WSA for optimal feature selection and data fusion

A. Dadras Eslamlou [a], A. Kaveh [b,*], M. Azimi [c], T.Y. Yang [c]

[a] School of Civil Engineering and Transportation, South China University of Technology, Guangzhou 510640, China
[b] School of Civil Engineering, Iran University of Science and Technology, Tehran-16, P.O. Box 16846-13114, Narmak, Iran
[c] Department of Civil Engineering, The University of British Columbia, Vancouver, BC V6T 1Z4, Canada

ARTICLE INFO

Keywords:
Feature selection
Algebraic WSA
Water strider algorithm
Structural health monitoring
Improved Dempster-Shafer theory

ABSTRACT

In this study, two binary versions of the Water Strider Algorithm (WSA) are proposed and applied to optimal feature selection in classification problems. In the new binary versions, the formulations of WSA in continuous space are converted into binary space using group-theoretic operators (in AWSA) and sigmoid function (in BWSA). AWSA, BWSA, genetic algorithm (GA), and binary particle swarm optimization (BPSO) are selected and compared over eighteen well-known datasets from the University of California, Irvine repository. The results of AWSA indicate its satisfactory performance compared to those of other algorithms. Then, they are applied to find optimal features of a structural health monitoring classification problem using two well-known machine learning classifiers, namely *k*-Nearest Neighbor (KNN) and Naïve Bayes (NB) algorithms. To further improve the accuracy of the classification models, a decision-level data fusion technique is proposed based on the improved Dempster-Shafer theory. It is demonstrated that the AWSA presents superior results compared to the other algorithms and the suggested decision-level data fusion provides a reliable detection of damage.

1. Introduction

Nowadays, due to the extensive use of sensors in industry and universal access to the Internet, an immense amount of data is collected. Machine learning (ML) algorithms allow researchers to utilize these data through high-performance computers and train efficient models for decision making in a wide variety of applications [62,21]. Data processing can be challenging and time-consuming when the collected data contains so many features [16]. Additionally, when some of these features are redundant, irrelevant, or inaccurate, the final model becomes complex and leads to lower accuracy and generalization [50]. In such cases, it is recommended to remove the redundant dimensions and train the model using the optimal features. Although increasing the number of ML parameters or training samples might also result in higher accuracy, the process would be time- and computation-intensive.

Many studies in structural health monitoring (SHM) developed and investigated damage detection techniques; however, the majority of them used model-based methods [11]. This method is based on recursive updating of a finite element model, but it should be noticed that constructing an accurate multivariate and nonlinear model for every structure is a demanding task [4]. On the other hand, model-free methods aim to extract damage-sensitive features without the need for establishing a finite element model of the structure. Execution time, accuracy, and robustness of ML models are of great importance in SHM, particularly in damage detection applications. Thus, it is crucial to select the optimal features for training ML models in a short time and reach high accuracy [48,47,14,55].

There are generally three categories for feature selection (FS), namely filter-based, embedded, and wrapper methods [64]. Filter-based methods use statistical scores, such as Fisher score [56], Pearson correlation [40], Chi-square [39], and gain ratio [49], to choose the most informative, or less-correlated features. Embedded methods are employed by learning algorithms, such as Lasso and Ridge regression [51,20], which possess built-in functions to select optimal features during the training phase. Wrapper methods [3] evaluate and score multiple subsets of features based on the performance of the ML algorithm to find the optimal combination to maximize the model performance.

Selecting the optimal features by wrapper method is an NP-hard optimization problem aiming at minimizing the number of features and increasing the accuracy of the final model [10,19]. In general, trying all subsets of features (brute-force search) is a computationally

* Corresponding author.
E-mail address: alikaveh@iust.ac.ir (A. Kaveh).

https://doi.org/10.1016/j.istruc.2023.105280
Received 27 February 2023; Received in revised form 16 September 2023; Accepted 19 September 2023
Available online 28 September 2023

demanding task, because for N number of features $2^N - 1$ subsets must be trained and scored, which is impractical for medium- and large-scale problems. Therefore, it is necessary to employ fast optimization algorithms to find the optimal or near-optimal features. The classic optimization methods have difficulty dealing with such discrete problems. For example, they get stuck in local minima, are sensitive to the initial position, and on top of that, their implementation is not an easy task because of lacking the mathematical formulation in most of the feature selection problems.

As an alternative solution, the application of Evolutionary Computation (EC) and Swarm Intelligence (SI) metaheuristic algorithms has gained growing popularity recently [29,1]. These stochastic algorithms, unlike the classic methods, offer feasible solutions in a reasonably short time and with acceptable accuracy. Metaheuristics are generally nature-inspired search algorithms that employ biological, physical, or behavioral patterns and mechanisms to make a balance between global search (exploration) and local search (exploitation) [25] and [5]. Among many available algorithms, Genetic Algorithm (GA), and Particle Swarm Optimization (PSO) have drawn significant attention from researchers in different applications. However, due to the No Free Lunch (NFL) theorem [57], many other new algorithms – such as Artificial Bee Colony (ABC) [24], Harmony Search [13], and Firefly Algorithm [61] – have also been developed and applied to FS problem in recent years. Most of these algorithms are originally developed for continuous problems, and the discretized versions of these algorithms may cause malfunctioning.

The most recent studies on the application of metaheuristics in FS can be generally divided into three categories:

The first category enhances the performance of metaheuristics by incorporating technical strategies. For example, Ouadfel and Abd Elaziz [46] incorporated three mechanisms including, an adaptive awareness probability, a dynamic local neighborhood, and a novel global search strategy in Crow Search Algorithm to alleviate its premature convergence; Kılıç et al. [34] integrated PSO with a multi-population strategy which starts with two sets of populations generated by random and Relieff-based initializations.

The second category combines two or more algorithms to take advantage of different search behaviors and thereby enhancing their search capabilities. For instance, Shunmugapriya and Kanmani [54] combined the characteristics of Ant Colony Optimization and ABC to remove the stagnation behavior of the searching ants and the time-consuming global search for initial solutions by the searching bees; and Hussain et al. [22] integrated Sine-Cosine Algorithm (SCA) in Harris hawks Optimization (HHO), to eliminate ineffective exploration in HHO, and enhance the exploitation through adjusting candidate solutions dynamically. Xu et al. [58] used a combination of techniques to tune the balance between the global and local search capabilities of the Moth-flame Optimization (MFO) algorithm, including a simulated annealing disturbance mechanism, two types of transfer functions, and an ensemble strategy. Lin et al. [37] developed a large-scale information fusion system for integrating closed high-utility patterns across multiple distributed databases. To reduce the number of items among transactional groups, and the computational time, they presented a new huybrid genetic k-means (HG-k-means) algorithm.

The third category tackles the FS problem as a multiobjective optimization with two conflicting objectives, i.e. minimizing the number of the selected features and simultaneously maximizing the classification accuracy. For example, Dong et al. [7] presented an improved NSGA III algorithm to optimize several criteria, including ranking loss, average precision, coverage, MicroF, Hamming loss, and MacroF. Han et al. [17] developed a novel FS algorithm based on multi-objective PSO (MOPSO) with adaptive strategies (MOPSO-ASFS) to minimize the number of selected features and classification error. Zhou et al. [65] incorporated a flexible cut-point mechanism in MOPSO to find the optimal Pareto solutions in terms of the classification error, a distance metric, and the ratio of feature selection. Lin et al. [38] developed a multi-objective model for mining closed high utility itemsets, which makes use of Spark's MapReduce frameworks. They applied the multi-objective k-means algorithm to categorize transactions according to the relationship to the frequency component.

It is noteworthy that one of the main causes for the under-performance of algorithms in the binary search space is that the transformation mechanisms deteriorate the algorithmic search patterns and heuristics [45]. This issue will be further discussed later in Section 3. Santucci et al. [52] recently introduced an algebraic framework to convert continuous metaheuristics into binary algorithms in an effective manner, preserving their fundamental search heuristics.

The authors have proposed a new metaheuristic called Water Strider Algorithm (WSA), inspired by the life cycle of water strider bugs, which leads to considerably good results in different engineering and mathematical problems. In this paper, an algebraic version of WSA (AWSA) is proposed based on the framework presented in Ref. [52], and to the best of our knowledge, this is the first work that examines the application of this framework in the FS problem. AWSA is firstly assessed through 18 benchmark datasets. Due to the decent performance and popularity of GA and BPSO, their performances are also reported and compared with the proposed algorithm. Next, the effectiveness of the method is validated in an experimental SHM problem with time-domain data. In this problem, two ML classifiers, namely k-Nearest Neighbors (KNN) and Naïve Bayes (NB) algorithms are used, and a decision-level data fusion technique based on Dempster–Shafer theory (DST) is employed to increase the accuracy and robustness of the results.

The remainder of the paper is organized as follows: Section 2 presents the standard WSA. The proposed algorithm is introduced in Section 3. The background theory of Dempster-Shafer's theory (DST) is explained in Section 4. Section 5 presents the assessment of the algorithms in terms of statistical measures and tests. Finally, in Section 6, a data fusion framework is provided based on the proposed algorithm for structural health monitoring problems.

2. Water strider algorithm

WSA is a swarm optimization algorithm inspired by the life cycle of water strider insects (Gerridae), and their territorial life, ripple communication, foraging, and mating behavior [30]. In this algorithm, the solutions are represented by the position of a group of water strider bugs that can stride and live on lakes and rivers. Due to WSA's efficient explorative and exploitative search mechanisms, it recently has been successfully applied to several problems in civil engineering, such as inverse damage detection, and optimum design of structures [27,32].

To mimic the birth stage of the water strider's life, WSA starts the optimization process by generating various random solutions scattered around the search space (lake). These solutions are evaluated by the cost function of the problem, which represents the abundance of the food in their positions. In a minimization problem, the less value the cost function has, the higher abundance it possesses. To form the territories, they are sorted based on their costs from the best to the worst solution. Afterward, they are divided into groups and the territories are established. This process is illustrated in Fig. 1 for twenty water striders that are divided into three territories.

In nature, the most abundant areas are usually occupied by the female water striders, which are called "optimum foraging-habitat users". Therefore, the best solutions are considered for females. Additionally, each territory includes only one mature male ("keystone") that mates with the females inside the territories. In Fig. 1, the keystones of the territories are distinguished with a red circle.

Mating and foraging are two other main steps of a water strider's life. In the mating step, the position of the keystone is updated using one of the Eq. (1) which is related to successful or unsuccessful mating. In nature, the keystone sends mating signals through ripple waves toward a female inside the territory, and the female answers by either acceptance or rejection signals. Here, the probability of acceptance is assumed 50%.

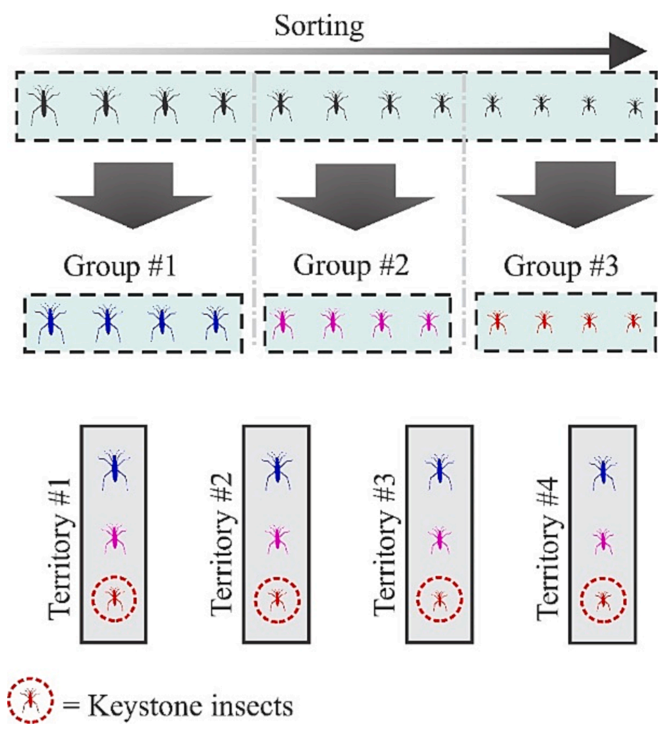


Fig. 1. Territory establishment process in WSA.

$$\begin{cases} \boldsymbol{WS}_i^{t+1} = \boldsymbol{WS}_i^t + \boldsymbol{R}.\boldsymbol{rand}_i & if mating\ happens(with probability of 50\%) \\ \boldsymbol{WS}_i^{t+1} = \boldsymbol{WS}_i^t + \boldsymbol{R}.(1+\boldsymbol{rand}_i) & otherwise \end{cases} \quad (1)$$

where vectors are denoted using boldface. $\boldsymbol{WS}_i^t$ is the position of the i^{th} water strider in the t^{th} cycle, and $\boldsymbol{rand}_i$ is the i^{th} random vector with values between 0 and 1. $\boldsymbol{R}$ denotes a vector starting at the keystone's position ($\boldsymbol{WS}_i^{t-1}$) and ends at the position of the target female ($\boldsymbol{WS}_F^{t-1}$). The target female is selected using roulette wheel selection among the resident females of the same territory.

Water striders spend a lot of energy in the mating step, which should be regained through food. The new position of the water strider should be evaluated by cost function – indicating the abundance of food, and if the new position is better than the previous one, it shows that it has access to enough food; otherwise, the keystone moves toward the best territory which hosts the most abundant position of search space (lake). This movement equation is formulated in Eq. (2)

$$\boldsymbol{WS}_i^{t+1} = \boldsymbol{WS}_i^t + 2\boldsymbol{rand}.(\boldsymbol{WS}_{BL}^t - \boldsymbol{WS}_i^t) \quad (2)$$

where $\boldsymbol{WS}_{BL}^t$ is the position of the best WS. It should be noted that the inability to improve the previous cost value shows that the water strider could not find food.

The resident water striders of the new territory usually behave aggressively against the intruders, and if the newcomer cannot find a better position than before – regarding the cost function value, it will be killed. In this case, another water strider (successor) will be generated randomly in the search space.

The territory establishment, mating, and feeding processes will be executed repeatedly for all territories until the termination condition of the algorithm is satisfied. In Algorithm 1, the pseudocode of WSA is presented.

2.1. Binary water strider algorithm

All steps of Binary WSA (BWSA) are the same as WSA, but the real-valued variables should be transformed into binary digits before calling the cost function for the evaluation of solutions. To achieve that, each variable is first transformed into a random variable with a value between 0 and 1 using the Sigmoid function Eq. (3), which takes the position of each water strider ($WS_{i,k}^t$) and returns the probability of taking 0 or 1 for each dimension (k).

$$T\left(v_{i,k}^t\right) = \frac{1}{1+e^{-WS_{i,k}^t}} \quad (3)$$

After the conversion, the binary positions of water striders are computed using the following operator:

$$P_{i,k}^t = \begin{cases} 0 & if rand < T(WS_{i,k}^t) \\ 1 & if rand \geq T(WS_{i,k}^t) \end{cases} \quad (4)$$

where *rand* is a random number between 0 and 1 with a uniform distribution. It is noteworthy that this should be generated in each iteration and for each variable of the water strider independently.

2.2. Time complexity analysis

In this section, the time complexity of the WSA as well as GA, PSO, and Ant Colony Optimization (ACO) algorithms in terms of their parameters are provided. In these algorithms, the computational time of the cost calculation is the dominating factor in processing time compared with other operations.

Three main processes should be considered when estimating the WSA's computational complexity. All N water striders must be randomly initialized and evaluated at the start of the process. As a result, the initialization operation has a computational complexity of $O(N)$. The attraction or struggling position updating occurs in the second phase for T number of territories, which equates to $O(T)$ complexity. Following the location update, the method may require 0, 1, or 2 assessments for each of the keystones. These assessments are carried out to find food or determine succession. In terms of solution quality, the computational complexity of the third phase lies between $O(0)$ and $O(2T)$. It is worth noting that the first process is applied only once, whereas the second and third processes are repeated multiple times (C cycles). Therefore, WSA possesses a computational complexity of between $O(N+C\times T)$ and at most $O(N+C\times 3T)$.

In GA, the initialization of N individuals results in computational complexity of $O(N)$. If m proportion of the individuals is reproduced as a new population, the algorithm results in $O(N+m\times N\times G)$, where G denotes the number of generations. With N particles and M iterations, PSO involves $O(N)$ complexity for random initialization and $O(N\times M)$ complexity for the main loop, resulting in $O(N+N\times M)$ complexity. Similarly, in ACO, having N searching ants and M iterations, the optimization process involves $O(N+N\times M)$ runtime complexity. In this study, to ensure a fair comparison between the algorithms, the maximum number of calls for the cost functions is set as the stopping criteria.

3. Algebraic water strider algorithm

In encoder-based binary versions of real-valued algorithms solutions are stochastically encoded into a binary string. This transformation can diminish the performance of the algorithms, since the heuristic notions behind the search moves, search neighborhood, and geometry of search space might become deteriorated. For example, a fixed real-valued numeric vector can be transformed into several quite different bit-strings with considerable distances in the binary space. Santucci et al. [52] proposed an algebraic framework for deriving discrete variants of such real-valued algorithms that tackle the mentioned issues mathematically. In the following sections, the concepts from group theory on which the framework is established are briefly introduced, and then, the counterpart formulations of WSA in this framework are provided.

3.1. Group theory preliminaries

Group theory is extensively used in structural analysis, e.g. see Kaveh and Nikbakht [31]. In the Algebraic WSA (AWSA), the binary search space is represented through an algebraic structure, or a group, with specific operators. Considering X as a set and $\star$ as a binary operator, a group $(X, \star)$ needs to possess the following three properties:

- *Identity element*: i.e., there exists a unique neutral element $e \in X$, such that for all $a \in X$, $a\star e = e\star a = a$.
- *Inverse element*: for every $a \in X$ there exists a unique inverse element a^{-1} from the same set, such that $a\star a^{-1} = a^{-1}\star a = e$.
- *Associativity*: for all $a, b, c \in X, a\star(b\star c) = (a\star b)\star c$.

In a finitely generated group $(X, \star)$, any $a \in X$ can be decomposed into a finite composition of generators $f_1, f_2, \cdots, f_l \in F$ where F (generating set) is a finite subset of X. In other words, the set a can be generated as $a = f_1\star f_2\star\cdots\star f_l$. It should be noted that every $a \in X$ can be produced by many decompositions with different lengths (l). In the next section, an algorithm for finding a minimal decomposition will be provided.

Every group can be represented with a Cayley digraph $C(X, \star, F)$ with vertices corresponding to the elements of X, and any pair of vertices (a, b) are connected with an edge f if and only if $b = a\star f$. In Fig. 2, a Cayley graph corresponding to a bit-string group with four bits is depicted. For an excellent explanation of graph theory, the reader may refer to Harary [18], and for extensive applications in structural analysis, one may refer to Kaveh [26].

3.2. Bit-string group

The set of the n-length bit-strings ($\mathbb{B}^n = \{0, 1\}^n$) forms a group with respect to the bitwise XOR operator, denoted by $\vee$. The properties of this group are presented below and proved in Appendix:

- *Identity element*

  A string with all-zero bits is defined as an identity element (e).

- *Inverse element*

  The inverse of an element is considered as itself, i.e., $a^{-1} = a$.

- *Associativity*

  For any a, b, $c \in \mathbb{B}$, $a \vee (b \vee c) = (a \vee b) \vee c$.

A suitable generating set for a binary group is the flip of bits from zero to one and vice versa, which is mathematically expressed by $U = \{u_i \in \mathbb{B}^n | u_i(i) = 1\&u_i(j) = 0 for j \neq i\}$. For example, considering $a = (10001)$ as a binary element and $u_3 = (00100)$ as a generator, $a \vee u_3$ results in the flip of the third bit of a as (10101). Moreover, the generating set of an element such as (10011) corresponds to $< u_1, u_4, u_5 >$, or any other combination of these three generators.

To implement this theory in metaheuristic algorithms, the group-theoretic counterparts of numerical addition ($\oplus$), subtraction ($\ominus$), and multiplication ($\odot$) operators should be developed. The addition operation $c = a \oplus b$ is defined as the application of the vector b (a path on the Cayley graph) to the point a (a vertex on the Cayley graph). Hence, by decomposing the vector b as $< f_1, f_2, \cdots, f_l >$, c is computed by $c = a\star(f_1\star f_2\star\cdots\star f_l)$. The subtraction operation $c = a \ominus b$ is defined as the difference between two vectors a and b. Hence, by decomposing the Hamming distance of these two elements as $< f_1, f_2, \cdots f_l >$, c is computed as $(f_1\star f_2\star\cdots\star f_l)$. To implement the scalar multiplication ($m \odot a$), two cases are defined; in the first case, the multiplier possesses a positive real value less than one ($m < 1$), and in the second case, it is equal to or higher than one ($m \geq 1$). In the first case, the weight of the element ($|a|$) is calculated as its Hamming weight. Afterward, a is decomposed into its basic generators with Hamming weight of 1. Eventually, a set of $[m.|a|]$ number of the generators should be randomly selected and applied to the identity element (e). In the second case, the difference between the maximal weight element (ω) filled with "1″ bits and the element a is calculated as explained above. Then, the calculated difference is decomposed into its basic generators. Finally, $[m.|a|] - 1$ generators are randomly selected and applied to the element a.

3.3. Algebraic formulations for AWSA

In the algebraic version of the WSA (AWSA), the equations presented in Section 2 are reformulated using algebraic operators. In the following, the new formulas are provided.

In the birth step, the water striders' position should be randomly generated in the binary space. Eq. (1) regarding the mating process is reformulated as Eq. (5)

$$\begin{cases} \boldsymbol{WS}_i^{t+1} = \boldsymbol{WS}_i^t \oplus rand \odot \boldsymbol{R} & if mating\ happens (with probability of 50\%) \\ \boldsymbol{WS}_i^{t+1} = \boldsymbol{WS}_i^t \oplus (1 + rand) & otherwise \end{cases} \tag{5}$$

where, all variables are the same as in the previous equation, but the parameter *rand* is a scalar random number between 0 and 1, instead of a random vector, and the $\boldsymbol{R}$ calculated as Eq. (6)

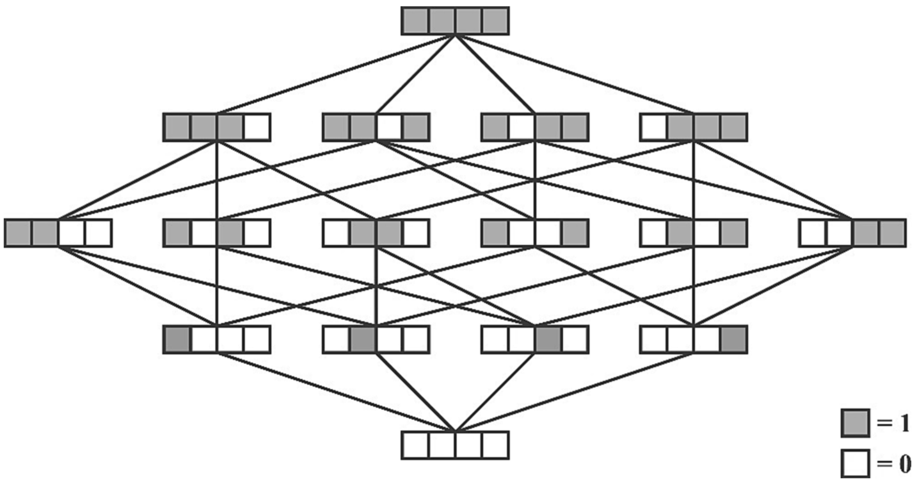


Fig. 2. An illustration of a Cayley graph.

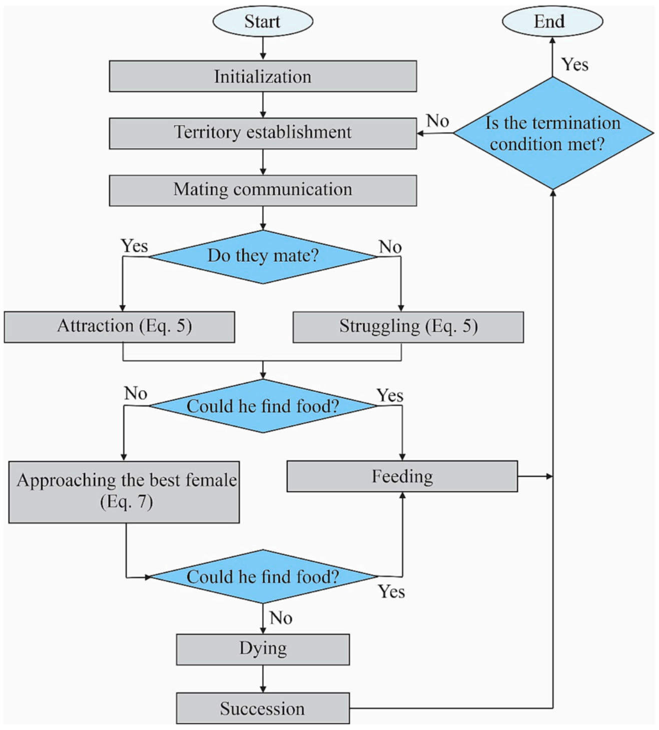


Fig. 3. The flowchart of AWSA.

$$\boldsymbol{R} = (\boldsymbol{WS}^t_{S,i} \ominus \boldsymbol{WS}^t_i) \tag{6}$$

in which, the $\boldsymbol{WS}^t_{S,i}$ is the selected female water strider for mating with i^{th} water strider in t^{th} cycle as described in Section 2. Additionally, the feeding process imitated by Eq. (2) is formulated as below

$$\boldsymbol{WS}^{t+1}_i = \boldsymbol{WS}^t_i \oplus 2rand.(\boldsymbol{WS}^t_{BL} \ominus \boldsymbol{WS}^t_i) \tag{7}$$

where, again the *rand* parameter is a uniform random number between 0 and 1.

The flowchart of the proposed algorithm is illustrated in Fig. 3.

4. Dempster-Shafer theory

The Dempster-Shafer theory (DST), or evidence theory, is a mathematical approach for data fusion considering different degrees of uncertainties in the process of classification, and incomplete information. It was originally proposed by Dempster [6], and further developed by Shafer [53]. This theory relies on three basic metrics, namely Basic Probability Assignment (BPA), Belief (BEL), and Plausibility (PL).

BPA, or mass function, is mathematically expressed as follows:

$$m : P \rightarrow [0, 1] : where\ m(\varnothing) = 0\ and\ \sum_{A \in P} m(A) = 1 \tag{8}$$

where P is the set of all possible subsets, $m(A)$ is the measure of the probability of A, and $\varnothing$ is the null set. *BEL* represents the total evidence to support an outcome, calculated as follows:

$$Bel(A) = \sum_{B \subseteq A} m(B) \tag{9}$$

PL, also called the upper bound, presents an uncertainty measure for the assumed outcome:

$$PL(A) = \sum_{B \cap A = \varnothing} m(B) \tag{10}$$

Dempster rule provides a formulation for combining two mass functions, such as m_1 and m_2, regarding two pieces of evidence by Eq. (11)

$$m(A) = m_1(B)m_2(C) = (1-K)^{-1} \sum_{B \cap C = A} m_1(B) \times m_2(C),\ when\ A \neq \varnothing \tag{11}$$

K is the mass associated with conflict which is determined by the summation of *BPA* product of sets with the null intersection as Eq. (12):

$$K = \sum_{B \cap C = \varnothing} m_1(B) \times m_2(C) \tag{12}$$

4.1. Improved Dempster-Shafer theory

Zadeh [63] criticized DST, pointing out paradoxical examples in which the Dempster rule leads to counterintuitive results. Yager [59] addressed the problem by classifying the conflicting evidence into the set Θ which presents the classifier's lack of knowledge and replaced *BPA* with ground probability assignment (q), which are formulated in the following:

$$\Theta_i = 1 - \alpha_i \tag{13}$$

$$q(A) = \sum_{\cap A_i = A} [m_1(A_1) \times m_2(A_2) \times \ldots \times m_i(A_i)] \tag{14}$$

where α_i is called the importance factor, or weight factor. Given an output (O_i), the new mass is calculated as

$$\mathrm{m}_i = \alpha_i \times O_i \tag{15}$$

In Yager's rule, contrary to Dempster's rule, the ground probability assignment of the null set can be larger than zero ($q(\varnothing) \geq 0$). Finally, the new combination rule is defined by Eq. (16):

$$m(A) = \frac{q(A)}{1 - q(\varnothing)} \tag{16}$$

In this study, the improved version of DST is applied to decision fusion in the SHM problem to reach more consistent, accurate, and robust classifications. Further details about these methods can be found in Refs. [60,53,35].

5. Experimental results and discussions

To investigate and compare the efficiency of optimizers, eighteen well-studied datasets from the UCI ML repository [12] are selected and implemented in this section. These datasets are listed in Table 1, and they cover a wide range of problems with various features and samples.

KNN ($k = 5$) is selected as the classification method, and the testing

Table 1
Machine Learning Benchmark datasets.

DS#	Dataset	Number of Features	Number of Samples
1	Tic-tac-toe	9	958
2	Vote	16	300
3	WaveformEW	40	5000
4	Wine	13	178
5	Zoo	16	101
6	BreastCancer	9	699
7	BreastEW	30	569
8	CongressEW	16	435
9	Exactly	13	1000
10	Exactly2	13	1000
11	HeartEW	13	270
12	Ionosphere	34	351
13	KrVsKpEW	36	3196
14	Lymphography	18	148
15	M−of−n	13	1000
16	PenglungEW	325	73
17	Sonar	60	208
18	SpectEW	22	267

Table 2
Parameter setting for algorithms.

GA	Number of Chromosomes	8
	Crossover Fraction	0.8
	Mutation Rate	0.02
BPSO	Number of Populations	8
	ω	Linearly decreased from 0.9 to 0.4
	c_1	2
	c_2	2
BWSA and AWSA	Number of Water striders	8
	Number of Territories	2

and training data are determined by 10-fold cross-validation to avoid overfitting. The objective of the optimization problem is defined as the minimization of classification error as well as the number of selected features, which is mathematically expressed as [41]:

$$Minimize : Cost = \gamma \times ER + (1 - \gamma) \times SR \tag{17}$$

where *ER* denotes the classification error rate, *SR* is defined as the ratio of the number of selected features to the number of all attributes, and γ is a control parameter that is assumed as 0.99 according to Ref. [9]. Furthermore, the internal parameters of the tested algorithms -AWSA, BWSA, GA [15], and BPSO [33] - are summarized in Table 2, and the maximum number of function evaluations ($NFE = 560$) is selected as the stopping criterion for all optimizers [9,42]. It is noteworthy that the search space of all algorithms is the same. Due to the stochastic nature of algorithms, each example is computed thirty times independently.

5.1. Statistical measures

In this section, the performance of the algorithms in terms of several statistical measures, namely minimum cost (Min), the average cost (Mean), standard deviation (Std), and average selection ratio (AvgSR) are compared. The results for the datasets are presented in Table 3, in

Table 3
The statistical results for different datasets.

DS #1					DS #2				
	Min	Mean	Std	AvgSR		Min	Mean	Std	AvgSR
AWSA	0.1628	**0.1677**	**0.0024**	0.8593	AWSA	**0.0382**	**0.0437**	**0.0029**	**0.2979**
BWSA	**0.1535**	0.1698	0.0077	**0.7852**	BWSA	0.0413	0.045	0.0038	0.3229
GA	0.1618	0.1721	0.0087	0.7926	GA	0.0388	0.0467	0.0056	0.3917
BPSO	0.1607	0.1732	0.0115	0.8148	BPSO	0.0388	0.0451	0.0033	0.3
DS #3					DS #4				
	Min	Mean	Std	AvgSR		Min	Mean	Std	AvgSR
AWSA	**0.1568**	**0.1635**	0.0033	0.5575	AWSA	**0.0324**	**0.0423**	0.0048	0.5051
BWSA	0.1588	0.1665	0.005	0.5667	BWSA	0.0372	0.0444	**0.0045**	0.5128
GA	0.1578	0.1692	0.0071	**0.555**	GA	**0.0324**	0.0431	0.0059	**0.4897**
BPSO	0.1646	0.1693	**0.0026**	0.5708	BPSO	**0.0324**	0.0424	0.0054	0.5179
DS #5					DS #6				
	Min	Mean	Std	AvgSR		Min	Mean	Std	AvgSR
AWSA	**0.0332**	**0.0362**	**0.0053**	**0.4229**	AWSA	**0.0265**	**0.0301**	0.0013	**0.6333**
BWSA	**0.0332**	0.0388	0.0067	0.4458	BWSA	**0.0265**	0.0303	0.0014	0.6667
GA	**0.0332**	0.0484	0.0149	0.4667	GA	0.0279	0.0305	0.0013	0.6593
BPSO	**0.0332**	0.0451	0.0079	0.4958	BPSO	0.0293	0.0312	**0.0011**	0.6556
DS #7					DS #8				
	Min	Mean	Std	AvgSR		Min	Mean	Std	AvgSR
AWSA	**0.0448**	**0.0484**	0.0019	0.3756	AWSA	0.0317	**0.0351**	**0.0023**	0.3125
BWSA	0.0472	0.0509	**0.0017**	**0.3578**	BWSA	0.0322	0.0365	**0.0023**	0.3458
GA	0.0465	0.0549	0.0063	0.4233	GA	**0.0311**	0.0375	0.0035	0.3083
BPSO	0.0458	0.0494	0.0024	0.3978	BPSO	0.0322	0.0385	0.0036	**0.2979**
DS #9					DS #10				
	Min	Mean	Std	AvgSR		Min	Mean	Std	AvgSR
AWSA	**0.0046**	**0.0046**	**0**	**0.4615**	AWSA	**0.2358**	**0.2402**	0.0007	**0.0897**
BWSA	**0.0046**	0.0459	0.1002	**0.4615**	BWSA	0.2397	0.2403	**0.0002**	0.1026
GA	**0.0046**	0.0246	0.0762	0.4692	GA	0.2363	0.2404	0.001	0.1436
BPSO	**0.0046**	**0.0046**	**0**	**0.4615**	BPSO	0.2379	**0.2402**	0.0006	0.1
DS #11					DS #12				
	Min	Mean	Std	AvgSR		Min	Mean	Std	AvgSR
AWSA	0.1374	0.1454	0.0056	0.4308	AWSA	0.0698	**0.0884**	0.0104	**0.2765**
BWSA	0.1403	0.1564	0.0082	**0.3949**	BWSA	**0.0638**	0.0956	0.0122	0.3265
GA	**0.1366**	**0.1437**	0.0052	0.4231	GA	0.0794	0.1015	0.0123	0.3441
BPSO	0.1387	0.1467	**0.0031**	0.4077	BPSO	0.0847	0.0992	**0.0072**	0.2784
DS #13					DS #14				
	Min	Mean	Std	AvgSR		Min	Mean	Std	AvgSR
AWSA	**0.017**	**0.0232**	**0.0028**	**0.5176**	AWSA	0.0931	**0.1173**	0.0166	0.513
BWSA	0.0239	0.0317	0.0177	0.5417	BWSA	**0.0925**	0.133	0.0191	0.5259
GA	0.0187	0.0306	0.017	0.5222	GA	**0.0925**	0.1335	0.0238	**0.5019**
BPSO	0.0199	0.0264	0.0033	0.5222	BPSO	0.0986	0.1263	**0.0159**	0.5037
DS #15					DS #16				
	Min	Mean	Std	AvgSR		Min	Mean	Std	AvgSR
AWSA	**0.0046**	**0.0046**	**0**	**0.4615**	AWSA	0.0718	**0.0817**	0.0095	**0.4446**
BWSA	**0.0046**	**0.0046**	**0**	**0.4615**	BWSA	0.0589	0.0886	0.0115	0.4474
GA	**0.0046**	**0.0046**	**0**	**0.4615**	GA	**0.0584**	0.0822	0.0118	0.4438
BPSO	**0.0046**	0.0098	0.0247	0.4846	BPSO	0.0586	0.0842	**0.0092**	0.4662
DS #17					DS #18				
	Min	Mean	Std	AvgSR		Min	Mean	Std	AvgSR
AWSA	0.0951	**0.1209**	0.0117	0.465	AWSA	**0.1371**	**0.1589**	0.0095	**0.4303**
BWSA	**0.081**	0.124	0.0156	**0.4561**	BWSA	0.1482	0.163	0.0092	0.4894
GA	0.0951	0.123	0.0135	0.4811	GA	0.1459	0.1613	0.0096	0.4348
BPSO	0.0997	0.1259	**0.0108**	**0.4561**	BPSO	0.1492	0.1674	**0.0079**	0.4409

Table 4
The results of statistical tests.

Problems	Comparison	*p*-values
DS #1	AWSA < BWSA < GA < BPSO	–
DS #2	AWSA < BWSA < BPSO < GA	–
DS #3	AWSA < BWSA < GA < BPSO	AWSA vs. GA: *p*-value = 0.00075559 AWSA vs. BPSO: *p*-value = 5.078e-06 BWSA vs. BPSO: *p*-value = 0.047447
DS #4	GA < AWSA < BPSO < BWSA	–
DS #5	AWSA < BWSA ≪ GA < BPSO	AWSA vs. GA: *p*-value = 3.3444e-05 AWSA vs. BPSO: *p*-value = 9.2095e-06 BWSA vs. GA: *p*-value = 0.011715 BWSA vs. BPSO: *p*-value = 0.0048154
DS #6	AWSA < BWSA < GA < BPSO	AWSA vs. BPSO: *p*-value = 0.015464
DS #7	AWSA < BPSO ≪ BWSA < GA	AWSA vs. BWSA: *p*-value = 0.00029252 AWSA vs. GA: *p*-value = 7.5496e-08 BWSA vs. BPSO: *p*-value = -0.018013 GA vs. BPSO: *p*-value = -2.8783e-05
DS #8	AWSA < BWSA < GA < BPSO	AWSA vs. GA: *p*-value = 0.0089445 AWSA vs. BPSO: *p* -value = 0.00046982
DS #9	AWSA = BPSO < GA < BWSA	AWSA vs. BWSA: *p* -value = 0.0041469 BWSA vs. BPSO: *p* -value = -0.0041469
DS #10	BWSA < BPSO < AWSA < GA	BWSA vs. GA: *p* -value = 0.040345
DS #11	GA < AWSA < BPSO ≪ BWSA	AWSA vs. BWSA: *p*-value = 1.3824e-07 BWSA vs. GA: *p*-value = -3.9878e-09 BWSA vs. BPSO: *p*-value = -0.00036802 GA vs. BPSO: *p*-value = 0.044801
DS #12	AWSA < BWSA < BPSO < GA	AWSA vs. GA: *p*-value = 0.00042193 AWSA vs. BPSO: *p*-value = 0.0013915
DS #13	AWSA ≪ BPSO < GA < BWSA	AWSA vs. BWSA: *p*-value = 1.0232e-07 AWSA vs. GA: *p*-value = 0.00016894 AWSA vs. BPSO: *p*-value = 0.0054247
DS #14	AWSA < BPSO < GA < BWSA	AWSA vs. BWSA: *p*-value = 0.0066756 AWSA vs. GA: *p*-value = 0.013418
DS #15	AWSA = BWSA = GA ≪ BPSO	AWSA vs. BPSO: *p*-value = 0.00071717 BWSA vs. BPSO: *p*-value = 0.00071717 GA vs. BPSO: *p*-value = 0.00071717
DS #16	AWSA < GA < BPSO < BWSA	–
DS #17	AWSA < GA < BWSA < BPSO	–
DS #18	AWSA < GA < BWSA < BPSO	AWSA vs. BPSO: *p*-value = 0.0046415 GA vs. BPSO: *p*-value = 0.046714

which the best results are written in boldface. As can be seen, considering the Min among all independent runs, the AWSA, BWSA, GA, and BPSO have reached the minimum cost in 11, 8, 8, and 4 cases, respectively. In terms of Mean, the algorithms AWSA, BWSA, GA, and PSO have obtained the best results in 17, 1, 2, and 2 cases, respectively; and in terms of AvgSR, AWSA in 9 datasets, and BWSA in 6 datasets have obtained the average minimum results. BPSO in 9 cases and AWSA in 7 cases have obtained the best Std among others; however, the mean values of AWSA are far better than BPSO as mentioned before. The results show that the proposed AWSA has a competitive performance compared to other powerful algorithms, regarding these statistical metrics,

5.2. *Statistical test*

In this section, a nonparametric statistical test, the so-called Kruskal-Wallis Test [36], is performed to compare the results of the algorithms. In this test, ranks of the data are used for comparison, rather than numeric values, and the *p*-value measures the significance of the chi-square statistic. In Table 4, for the sake of brevity, only the cases rejecting the null hypothesis of equal medians at a 5% significance level are reported. In the second column, the algorithms are rated from the best to worst algorithms, where '<' and '≪' signs stand for the nonsignificant and significant outperformance, and '=' stands for equal performance. In the last column, positive *p*-values show that the first-mentioned algorithm has lower costs than the second one and vice versa.

According to the rating in Table 4, in datasets #1, #2, #4, #16, and #17, none of the algorithms can reject the null hypothesis. AWSA has the best or equal performance compared to other algorithms in all problems, except for DS #4 and DS #11. AWSA outperforms BWSA, GA, and BPSO, in 5, 7, and 9 datasets, respectively; and it has never been outperformed by others at a 5% significance level. Additionally, BWSA, as the second-best algorithm in terms of mean rank, significantly outperforms BPSO or GA in only 4 datasets. All in all, this nonparametric test confirms that the results obtained by AWSA are generally better than others.

5.3. *Convergence behavior*

The convergence curve is another indicator by which the convergence speed and the local minima can be identified. The average convergence curves of the algorithms are presented in Fig. 4, in which the average cost values, computed by Eq. (17) for all runs, are plotted against the number of function evaluations (NFEs). While BPSO has the highest convergence rate in almost half of the cases, it usually gets stuck and has a premature convergence toward local minima. On contrary, although the AWSA has a high initial rate in three cases (datasets #1, #5, #6), it eventually has better performance than others, as seen in Fig. 4 and was proved in the preceding sections. Comparing the convergence curves of AWSA and BWSA, it becomes evident that the algebraic operators enhance the exploration behavior of the algorithm by which more optimal results are discovered, as well as the exploitation search which leads to constant improvement in the cost value in the course of iterations.

6. Application to structural health monitoring

6.1. *Experimental data*

The experimental vibration response data from the IASC-ASCE SHM benchmark building was used to validate the proposed framework [2,23]. As shown in Fig. 5, this benchmark model is a four-story quarter-scale steel frame that is 2.5 m × 2.5 m × 3.6 m. The force input to the structure was provided through ambient vibration, and the diagonal bracing members were removed to simulate different damage scenarios. In total, sixteen accelerometers were placed to measure the vibration response of the building in both horizontal directions. Nine damage patterns were simulated to investigate the performance of damage detection algorithms, which are as follows: (1) fully braced undamaged frame; (2) missing all east side braces; (3) removed braces on all floors in one bay on SE corner; (4) removed braces on 1st and 4th floors in one bay on SE corner (5) removed braces on 1st floor in one bay on SE corner; (6) removed braces on all floors on the east face, and the 2nd-floor braces on the north face; (7) all braced removed on all faces; (8) configuration 7, plus loosened bolts on all floors - both ends of beams on the east face, north side; (9) configuration 7, plus loosened bolts on floors 1 and 2 - both ends of beams on the east face, north side. The details of the experimental setup as well as the data can be accessed from the NEES Database for Structural Control and Monitoring Benchmark [8]_ENREF_1.

6.2. *Preprocessing and feature extraction*

Time series recorded for different damage patterns are preprocessed, and the trends and outliers are removed. These outliers could adversely affect statistical features which usually manifest themselves in undesirable performance of the classifiers. Afterward, 15 features in the time series listed in Table 5 are extracted from the recorded data, where *N* is

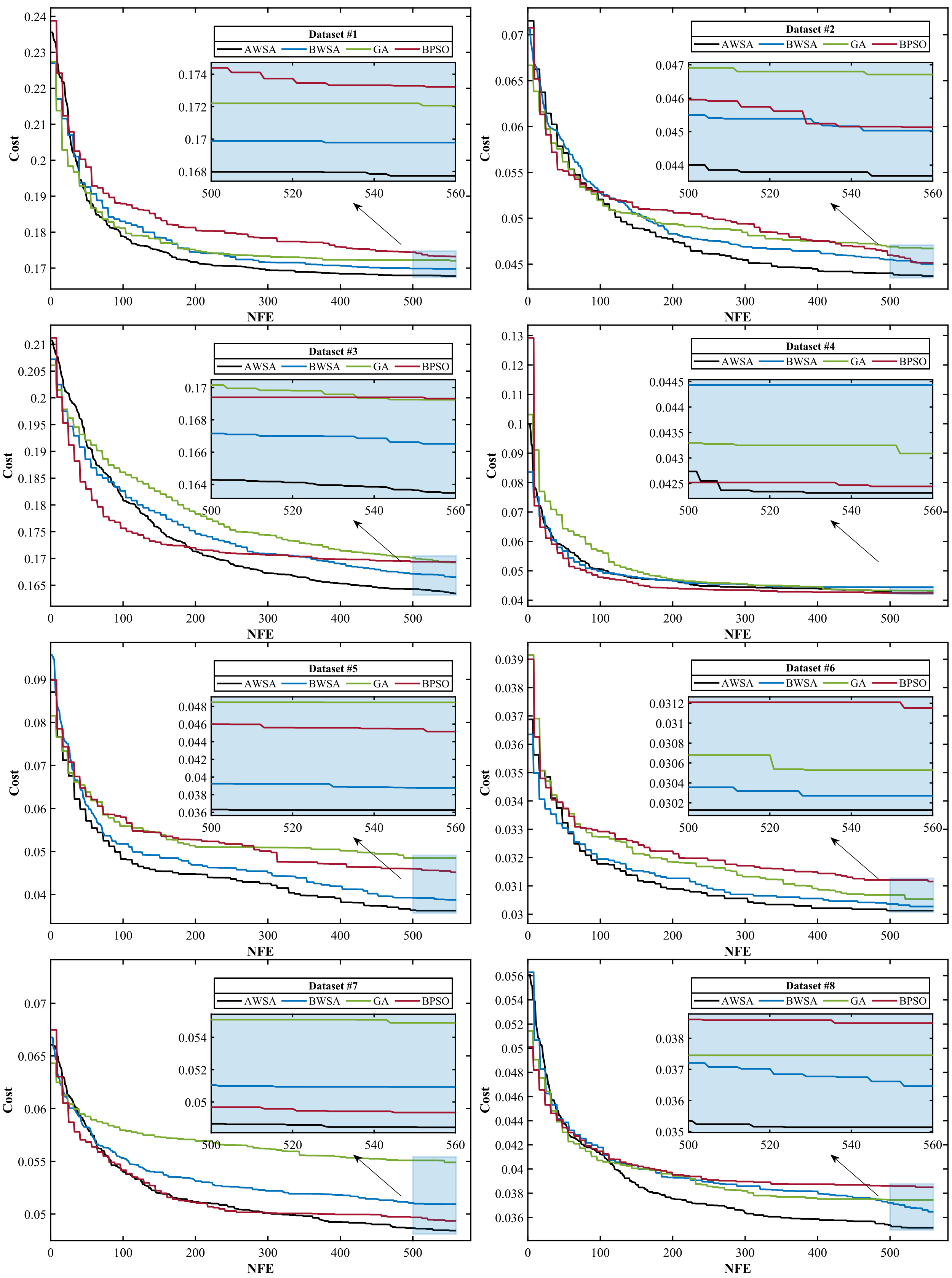


Fig. 4. The convergence curves of the algorithms for different benchmark problems.

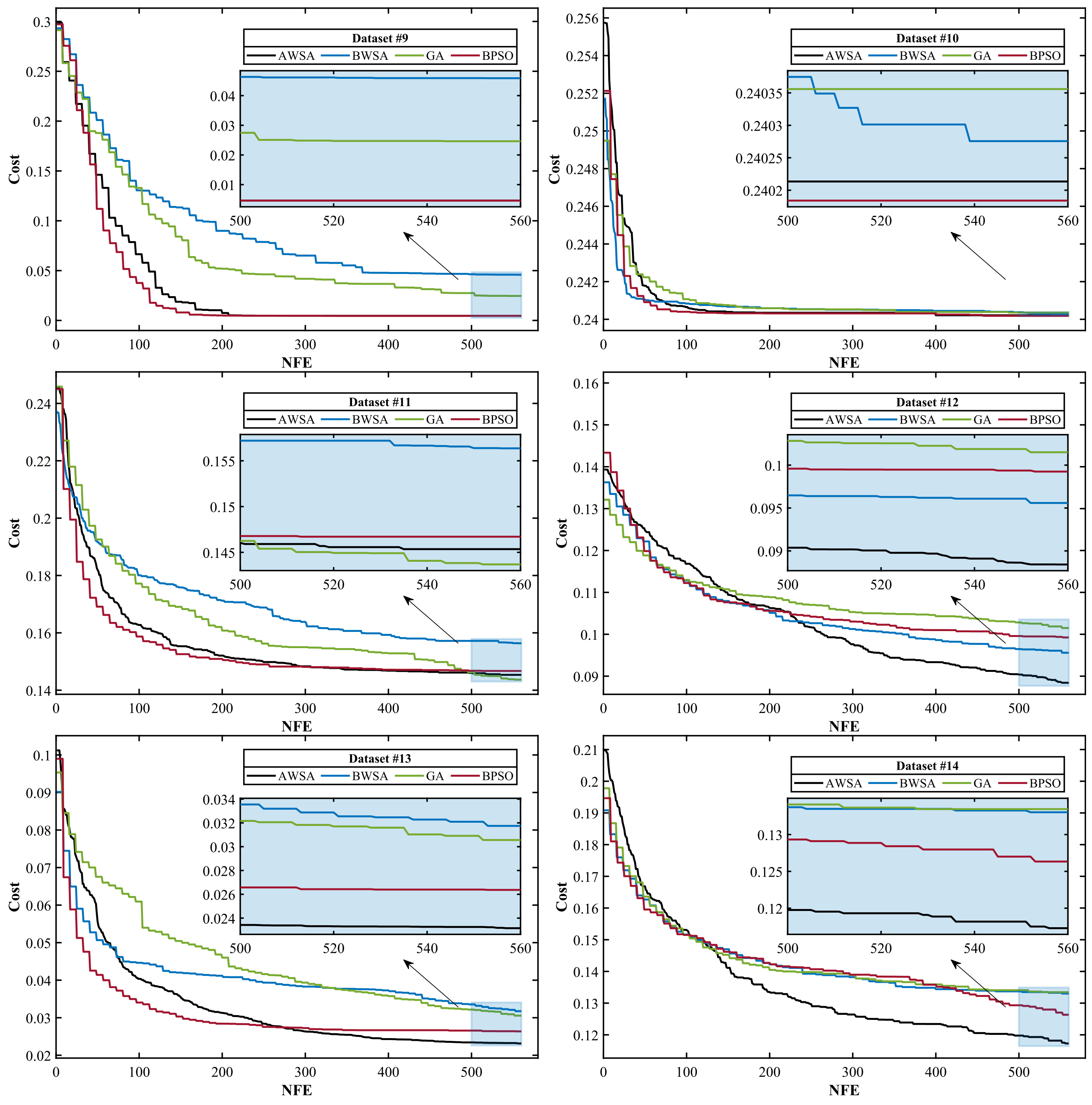


Fig. 4. (*continued*).

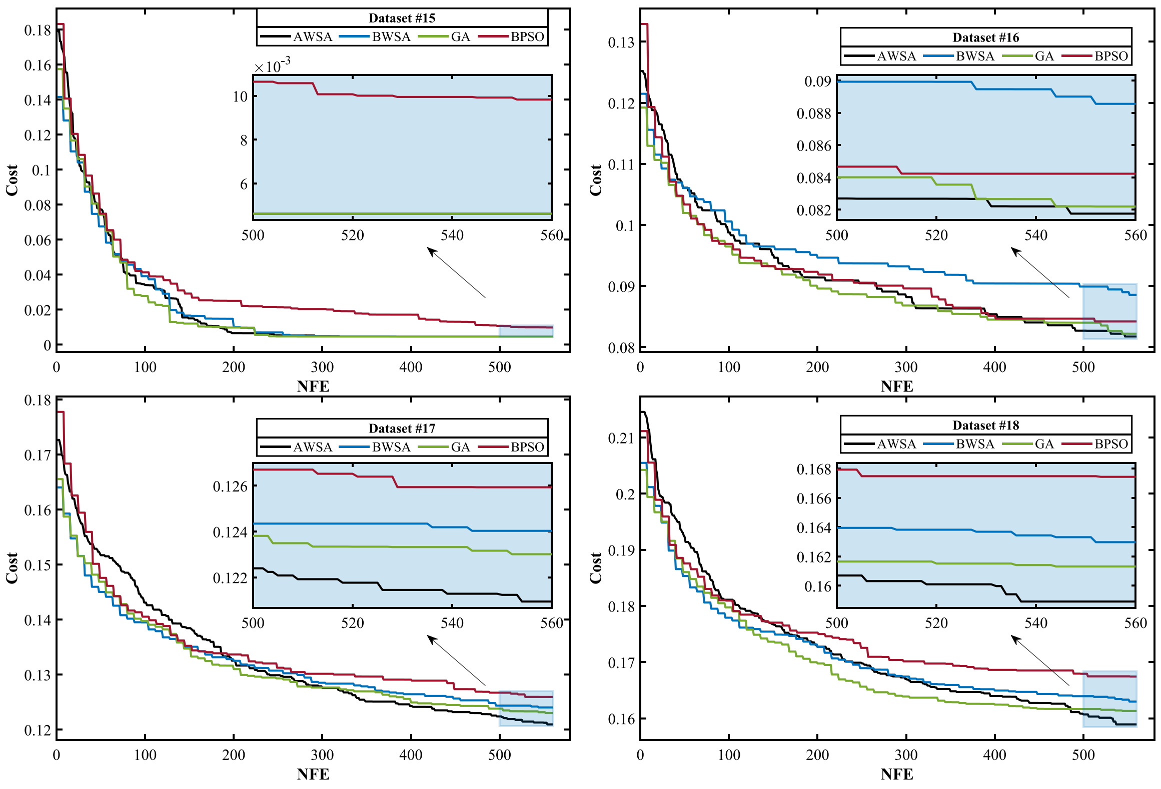


Fig. 4. (*continued*).

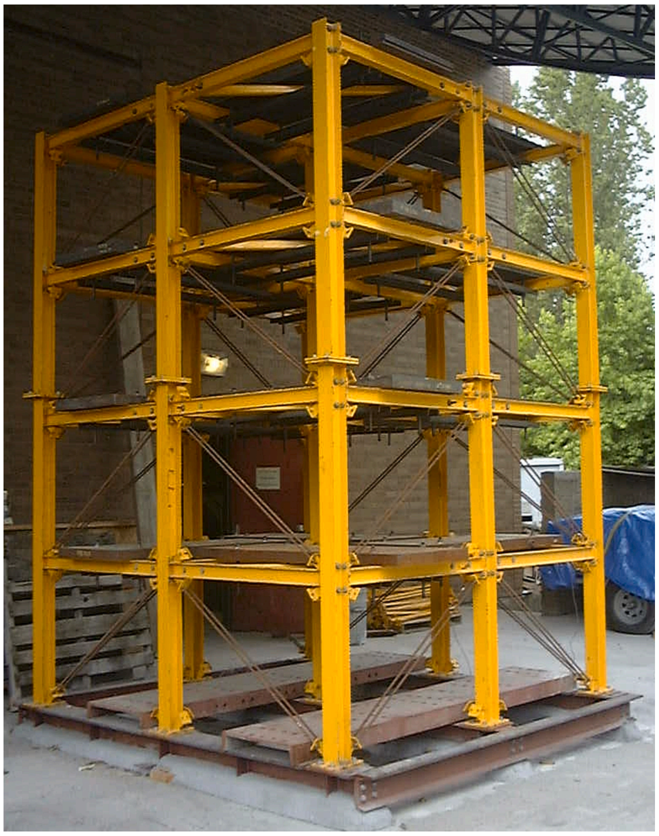


Fig. 5. The IASC-ASCE SHM benchmark (photo courtesy of Prof. Carlos Ventura, UBC).

Table 5
The features derived from time series.

Feature	Definition	Feature	Definition
Mean	$F_1 = \frac{\sum_{n=1}^{N} t(n)}{N}$	Impulse factor	$F_9 = \frac{F_2}{\frac{1}{N}\sum_{n=1}^{N} \lvert t(n)\rvert}$
Max	$F_2 = \max(\lvert t(n)\rvert)$	Crest factor	$F_{10} = \frac{F_2}{F_3}$
Root mean square	$F_3 = \sqrt{\frac{\sum_{n=1}^{N} (t(n))^2}{N}}$	3rd central moment	$F_{11} = \frac{\sum_{n=1}^{N} (t(n) - F_1)^3}{N-1}$
Standard deviation	$F_4 = \sqrt{\frac{\sum_{n=1}^{N} (t(n) - F_1)^2}{N-1}}$	4th central moment	$F_{12} = \frac{\sum_{n=1}^{N} (t(n) - F_1)^4}{N-1}$
Variance	$F_5 = \frac{\sum_{n=1}^{N} (t(n) - F_1)^2}{N-1}$	5th central moment	$F_{13} = \frac{\sum_{n=1}^{N} (t(n) - F_1)^5}{N-1}$
Skewness	$F_6 = \frac{\sum_{n=1}^{N} (t(n) - F_1)^3}{(N-1){F_4}^3}$	6th central moment	$F_{14} = \frac{\sum_{n=1}^{N} (t(n) - F_1)^6}{N-1}$
Kurtosis	$F_7 = \frac{\sum_{n=1}^{N} (t(n) - F_1)^4}{(N-1){F_4}^4}$	FM4	$F_{15} = \frac{F_{12}}{(F_5)^2}$
Shape factor	$F_8 = \frac{F_3}{\frac{1}{N}\sum_{n=1}^{N} \lvert t(n)\rvert}$		

the number of all data points in the time series t. It is noteworthy that other sophisticated features, such as frequency domain or time–frequency features, also could be used, but since the extraction of such features is computationally expensive, and to show the efficiency of the method, they are not included [44,43].

Table 6
The statistical results for KNN and NB classifiers.

Classifier	Algorithm	Min. Cost	Avg. Cost	Min. SR	Avg. SR
KNN	AWSA	**0.0387**	**0.0571**	**0.1333**	**0.2778**
	BWSA	**0.0387**	0.0893	0.2000	0.4844
	GA	**0.0387**	0.1657	0.2000	0.5711
	BPSO	**0.0387**	0.1438	0.2000	0.5133
NB	AWSA	**0.0167**	**0.0183**	**0.1333**	**0.2111**
	BWSA	0.0198	0.0287	0.2000	0.3467
	GA	**0.0167**	0.0197	0.2000	0.2189
	BPSO	**0.0167**	0.0190	0.2000	**0.2111**

Table 7
The results of Kruskal-Wallis test for KNN and NB classifiers.

Classifier	Comparison	P-values less than 0.05
KNN	AWSA ≪ BWSA < BPSO < GA	AWSA vs. BWSA: p-value = 0.019139 AWSA vs. GA: p-value = 8.369e-07 AWSA vs. BPSO: p-value = 0.00027389
NB	AWSA < BPSO < GA ≪ BWSA	AWSA vs. BWSA: p-value = 3.7757e-09 BWSA vs. GA: p-value = -3.7683e-09 BWSA vs. BPSO: p-value = -1.5706e-08

6.3. Optimal feature selection for KNN and NB classifiers

In this section, both KNN and NB algorithms are utilized for damage classification in the aforementioned SHM problem. The statistical results of 30 independent runs for the algorithms are presented in Table 6, whose best results are written in boldface. As seen, considering the KNN classifier, AWSA obtained the best results for all metrics, showing an efficient performance in reaching quality results. Although, in this case, all algorithms were able to find the minimum cost over thirty independent runs, AWSA's average cost is considerably less than the other's. Additionally, for the NB classifier, AWSA ranked first among others in terms of average cost, minimum SR, and average SR. Comparing the results of KNN to those of NB, it is evident that NB in this problem generally reaches lower cost values.

The results of the Kruskal-Wallis test feature selection with a p-value less than 0.05 for both KNN and NB classifiers are provided in Table 7. As seen, using the KNN classifier, AWSA outperformed the other algorithms, and in the case of the NB classifier, it works significantly better than BWSA. In both cases, AWSA has generally better performance than others.

The average convergence curves using KNN are plotted in Fig. 6(a), which shows the higher convergence rate of the AWSA compared with other algorithms. The convergence curves for NB are depicted in Fig. 6 (b), and, as can be seen, BPSO converges rapidly, yet AWSA converged to solutions with lower cost values.

The obtained optimal features are presented in Table 8. As shown, the second and fourth features (F_2 and F_4) are constantly among the optimum features for KNN, and F_2 is among the optimum features for NB.

Fig. 7 compares the confusion matrices of the sum of test results for KNN and NB algorithms trained by the optimal solutions of AWSA with those trained by all 15 features. This figure justifies the necessity of optimal feature selection for achieving better performance. For example, 37.8% of samples related to the fourth damage pattern are wrongly categorized by the KNN algorithm, and 20% of samples regarding the intact model are misclassified as damaged when all features are fed to NB. Therefore, feeding more features as the input data of classification algorithms can negatively impact the results. In the subsequent section, an innovative idea based on decision fusion is suggested which can enhance the metrics by combining the classification results of KNN and NB methods.

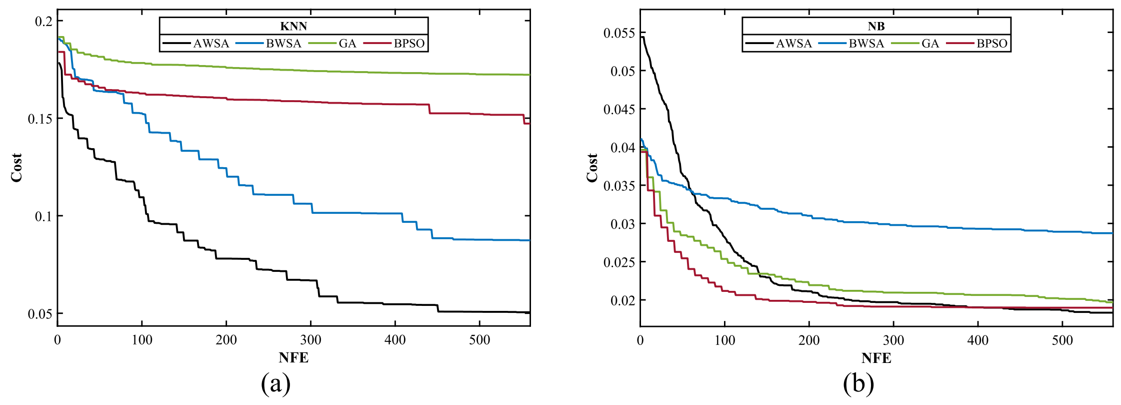
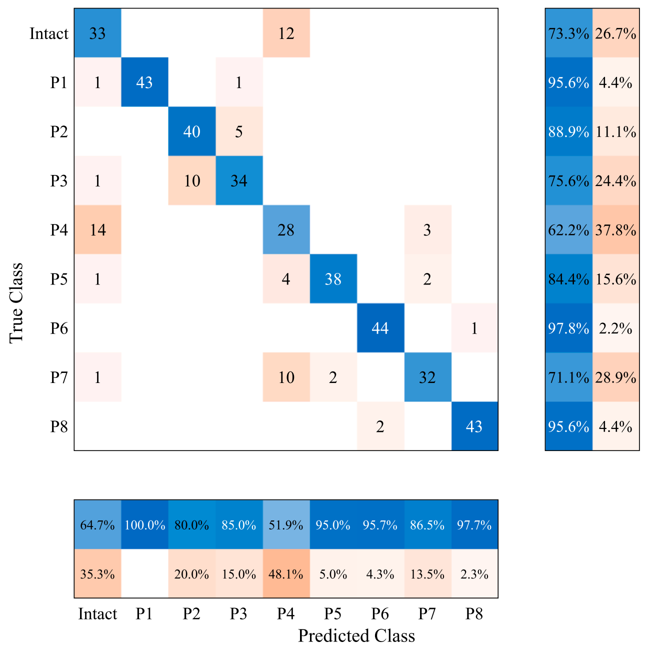


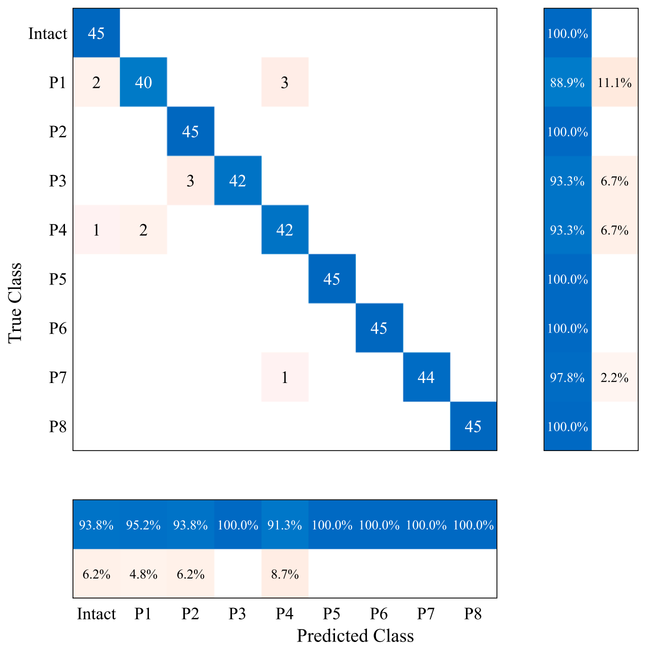
Fig. 6. The average convergence curves for (a). KNN, (b). NB.

Table 8
Optimum features for KNN and NB.

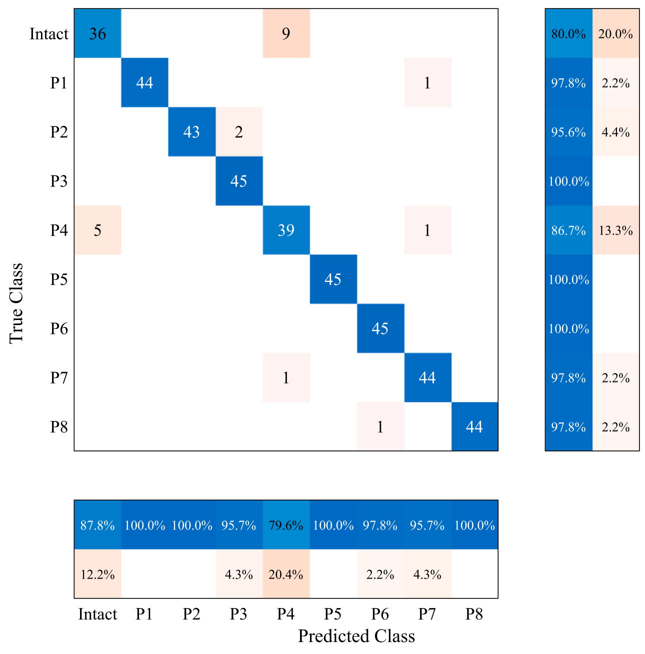
		Features														
	Algorithms	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15
KNN	AWSA		✓		✓								✓			
	BWSA		✓	✓	✓	✓						✓				
	GA		✓		✓								✓			
	BPSO		✓		✓							✓		✓		
NB	AWSA		✓		✓			✓								
	BWSA		✓		✓			✓								✓
	GA		✓			✓										✓
	BPSO	✓	✓	✓												✓



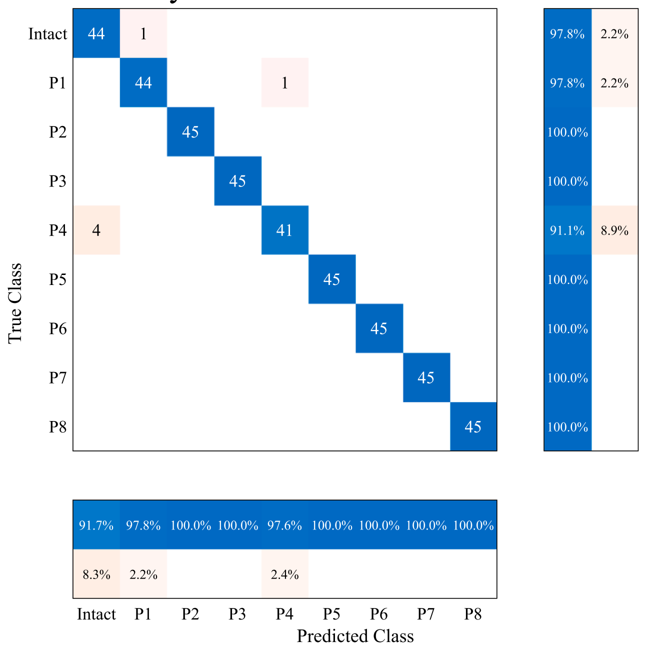
(a) KNN trained by all features



(b) KNN trained by the optimal features have been found by AWSA



(c). NB trained by all features



(d) NB trained by the optimal features have been found by AWSA

Fig. 7. Confusion matrices of KNN and NB.

6.4. *Decision fusion*

In this step, different classifiers are fused in the decision level using improved DST explained in Section 4.1. Since KNN and NB are among the fastest classifiers, the optimization and training process of these methods can be executed in a reasonably fast time, although their simple structure might cause low accuracy. Therefore, herein the output coming from these classifiers is integrated to determine the damage patterns accurately. The flowchart of the implemented decision-level fusion is illustrated in Fig. 8. As shown, this technique has a parallel structure which can be done within a short time. For this purpose, the best features found by AWSA obtained in the previous section using KNN and NB are utilized, whose α factors (weight factors) are assumed equal to 0.97, and 0.985, respectively.

The confusion matrix of the fused model is represented in Fig. 9(a), and its corresponding metrics are provided in Fig. 9(b). These figures show that decision fusion, having only one wrongly classified item, considerably improves classification. The classification metrics are also near one that is the best possible value.

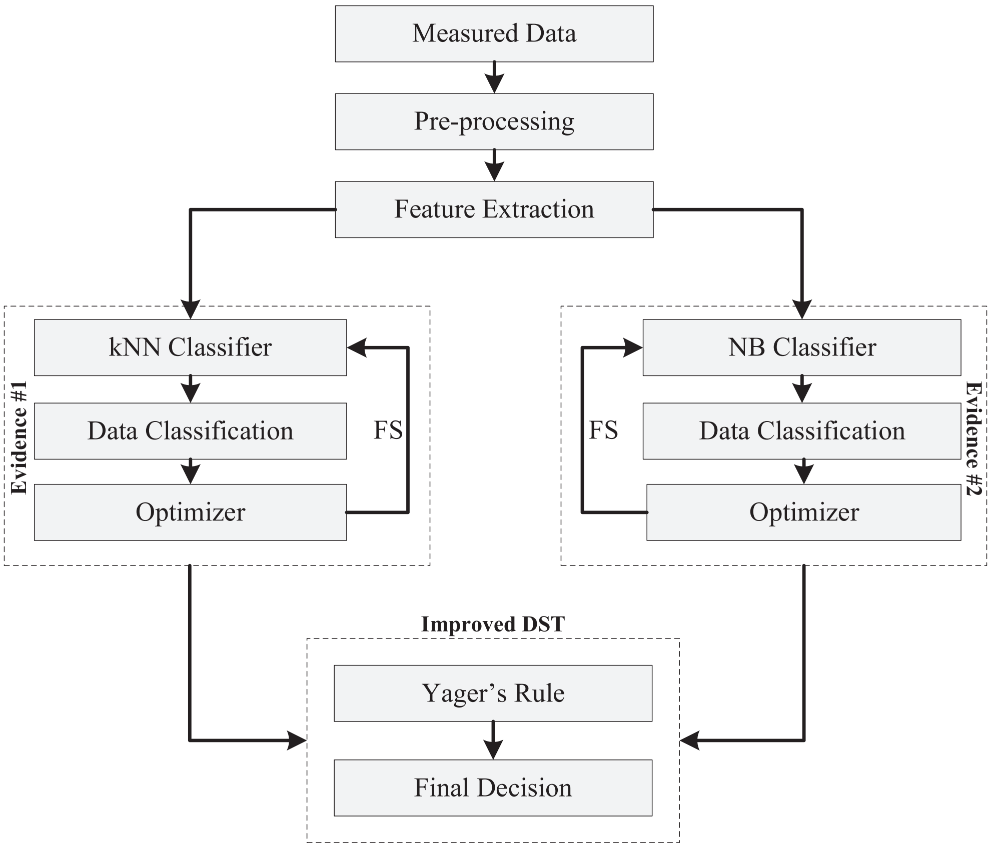


Fig. 8. The flowchart of the proposed decision fusion technique.

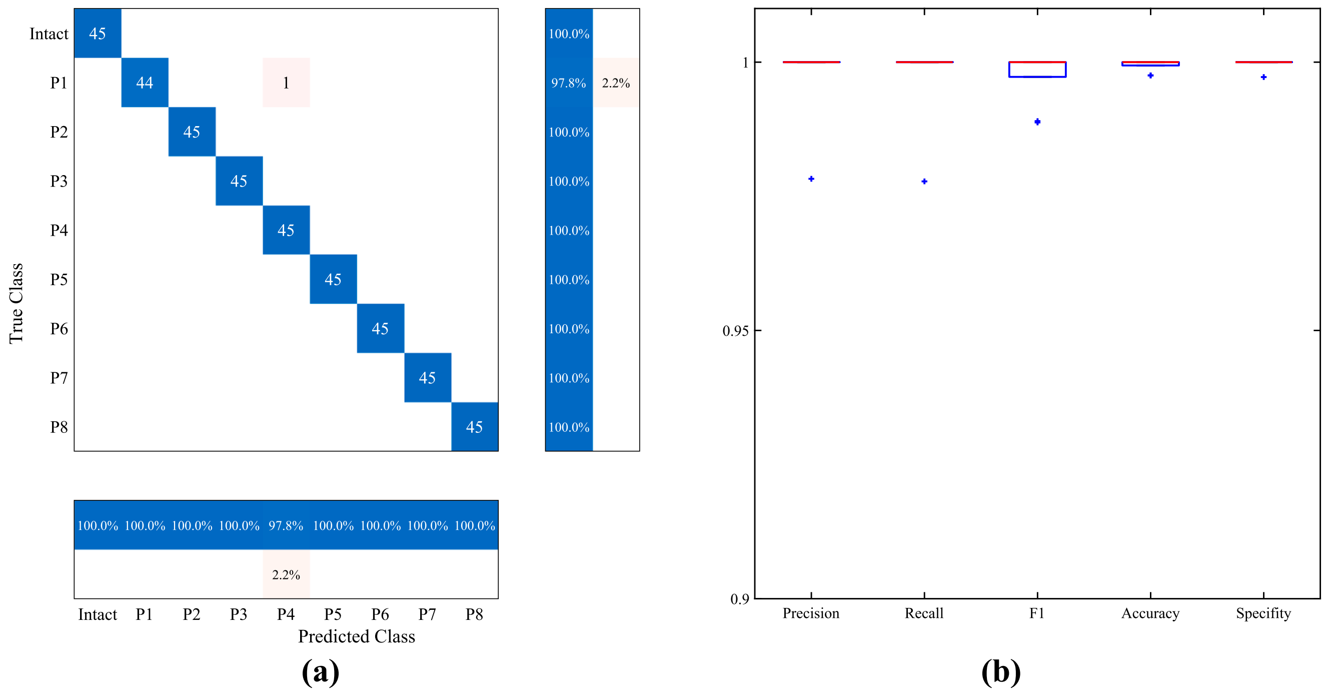


Fig. 9. The results of decision fusion: (a) confusion matrix, (b) classification metrics.

7. Conclusions

In this paper, an efficient algebraic version of the WSA algorithm (AWSA) has been introduced for feature selection in the wrapper method. Group theoretic binary counterparts of the formulations in continuous space have been developed for an efficient search to find the optimal subset of features. Eighteen datasets from the UCI repository have been selected as the first set of problems. The results show that the new algorithm generally outperforms other algorithms – such as BWSA, BPSO, and GA – in terms of statistical metrics and tests. The convergence behavior curves indicate that BPSO converges rapidly in the initial stages of optimization, yet AWSA usually finds low-cost solutions.

In the second part of the manuscript, the metaheuristic algorithms have been applied to an experimental SHM problem, where the new approach is capable of detecting damage patterns with acceptable accuracies. The improved DST has been suggested to fuse final decisions obtained by KNN and NB methods. This decision fusion technique indicates a further enhancement in the accuracy of the classification results.

With the recent rapid extensions of ANNs, Machine Learning and Altificial Intellegance, a great extension of the ideas like those presentd in this article is expected, Kaveh et al. [28].

The authors have applied and investigated the efficiency of the proposed algorithm to SHM problems. It can be utilized to tackle other combinatorial optimization problems, such as the knapsack problem.

Author contributions

All authors contributed equally to the study.

Funding

The study was not supported by external funding.

10. Availability of data and material

The data used in this study are available in UCI machine learning repository at https://archive.ics.uci.edu/ml/datasets.php and experimental phase II of the SHM benchmark problem dataset at https://bit.ly/3raZ5kw.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix

Definition 1. *Logical NOT is denoted by* $\neg$ *sign, such that*: $\neg 0 = 1$, *and* $\neg 1 = 0$.

Lemma 1. *For every* $a \in \mathbb{B}$, *we have* $a \vee 1 = \neg a$.

Proof: Let's consider any $a \in \mathbb{B}$. We have two cases:

Case 1. $a = 0$. *Then.*

$$a \vee 1 = 0 \vee 1 = 1 = \neg 0 = \neg a \tag{18}$$

Case 2. $a = 1$. *Then.*

$$a \vee 1 = 1 \vee 1 = 0 = \neg 1 = \neg a \tag{19}$$

In both cases, we see that $a \vee 1 = \neg a$, as required.

Lemma 2. *For every* $a, b \in \mathbb{B}$, *we have.*$a \vee \neg b = \neg(a \vee b)$

Proof: Let's consider any $a, b \in \mathbb{B}$. We have two cases:

Case 1. $b = 0$. *Then.*

$$a \vee \neg b = a \vee \neg 0 = a \vee 1 \tag{20}$$

and according to Lemma 1:

$$a \vee 1 = \neg a = \neg(a \vee 0) = \neg(a \vee b) \tag{21}$$

Case 2. $b = 1$. *Then.*

$$a \vee \neg b = a \vee \neg 1 = a \vee 0 = a = \neg(\neg a) \tag{22}$$

and according to Lemma 1:

$$\neg(a \vee 1) = \neg(a \vee b) \tag{23}$$

In both cases, the first term $a \vee \neg b$ leads to the second term $\neg(a \vee b)$ as required.

Proof of the identity element of the bit-string group:

Let's consider two possible cases for one bit of element a:

Case 1: $a = 0$. Then, it can be said

$$a \vee e = 0 \vee e = 0 \vee 0 = 0 \tag{24}$$

Case 2: $a = 1$. Then it means

$$a \vee e = 1 \vee e = 1 \vee 0 = 1 \tag{25}$$

Hence, it can be seen the defined identity element functions as a neutral element for both cases.

Proof of the inverse element of the bit-string group:

Let's consider the following two possible cases for a bit of a:

Case 1: $a = 0$. Then $a^{-1} = 0$, and it can be written as

$$a \star a^{-1} = 0 \vee 0 = 0 \tag{26}$$

Case 2: $a = 1$. Then $a^{-1} = 1$, and it means

$$a \star a^{-1} = 1 \vee 1 = 0 \tag{27}$$

The defined inverse element is valid because $a \star a^{-1}$ leads to e for both cases.

Proof of the associativity of the bit-string group:

To prove $a \vee (b \vee c) = (a \vee b) \vee c$ for any $a, b, c \in \mathbb{B}$, let's consider two cases:

Case 1. $c = 0$. *Then it yields.*

$$a \vee (b \vee c) = a \vee (b \vee 0) \tag{28}$$

Since 0 is the identity element:

$$a \vee (b \vee 0) = a \vee b = (a \vee b) \vee 0 = (a \vee b) \vee c \tag{29}$$

Case 2. $c = 1$. *Then.*

$$a \vee (b \vee c) = a \vee (b \vee 1) \tag{30}$$

using Lemma 1:

$$a \vee (b \vee 1) = a \vee \neg b \tag{31}$$

using Lemma 2:

$$a \vee \neg b = \neg (a \vee b) \tag{32}$$

then, using Lemma 1:

$$\neg (a \vee b) = (a \vee b) \vee 1 = (a \vee b) \vee c. \tag{33}$$

As seen, both cases result in the second term, $(a \vee b) \vee c$.

References

[1] Azimi M, Yeznabad AM. Swarm-Based Parallel Control of Adjacent Irregular Buildings Considering Soil-Structure Interaction. J Sens Actuator Netw 2020;9(2):18.

[2] Caicedo JM, Dyke SJ, Johnson EA. Natural excitation technique and eigensystem realization algorithm for phase I of the IASC-ASCE benchmark problem: Simulated data. J Eng Mech 2004;130(1):49–60.

[3] Chandrashekar G, Sahin F. A survey on feature selection methods. Comput Electr Eng 2014;40(1):16–28.

[4] De Roeck G. Model-Based Methods of Damage Identification of Structures Under Seismic Excitation. In: Limongelli MP, Çelebi M, editors. Seismic Structural Health Monitoring: From Theory to Successful Applications. Cham: Springer International Publishing; 2019. p. 237–59. https://doi.org/10.1007/978-3-030-13976-6_10.

[5] Del Ser J, Osaba E, Molina D, Yang X-S, Salcedo-Sanz S, Camacho D, et al. Bio-inspired computation: where we stand and what's next. Swarm Evol Comput 2019;48:220–50.

[6] Dempster AP. Upper and lower probabilities induced by a multivalued mapping. In: Yager RR, Liu L, editors. Studies in Fuzziness and Soft ComputingClassic Works of the Dempster-Shafer Theory of Belief Functions. Berlin, Heidelberg: Springer Berlin Heidelberg; 2008. p. 57–72.

[7] Dong H, Sun J, Sun X, Ding R. A many-objective feature selection for multi-label classification. Knowl-Based Syst 2020;208:106456. https://doi.org/10.1016/j.knosys.2020.106456.

[8] Dyke S (2011) Report on the Building Structural Health Monitoring Problem Phase 1 Experimental.

[9] Emary E, Zawbaa HM, Hassanien AE. Binary grey wolf optimization approaches for feature selection. Neurocomputing 2016;172:371–81.

[10] Faris H, Heidari AA, Al-Zoubi AM, Mafarja M, Aljarah I, Eshtay M, et al. Time-varying hierarchical chains of salps with random weight networks for feature selection. Expert Syst Appl 2020;140:112898.

[11] Firouzi B, Abbasi A, Sendur P. Improvement of the computational efficiency of metaheuristic algorithms for the crack detection of cantilever beams using hybrid methods. Engineering Optimization:1–22 2022;54(7):1236–57.

[12] Frank A, Asuncion A. UCI Machine Learning Repository [http://archiveicsuciedu/ml] 2010;213::2 .2.. School of information and computer science.

[13] Geem ZW, Kim JH, Loganathan GV. A new heuristic optimization algorithm: harmony search. Simulation 2001;76(2):60–8.

[14] Gharehbaghi VR, Farsangi EN, Yang T, Hajirasouliha I. Deterioration and damage identification in building structures using a novel feature selection method. In: Structures. Elsevier; 2021. p. 458–70.

[15] Goldberg DE. Genetic algorithms. Pearson Education India; 2006.

[16] Hamed A, Nassar H. Efficient feature selection for inconsistent heterogeneous information systems based on a grey wolf optimizer and rough set theory. Soft Comput 2021;25(24):15115–30. https://doi.org/10.1007/s00500-021-06375-z.

[17] Han F, Chen W-T, Ling Q-H, Han H. Multi-objective particle swarm optimization with adaptive strategies for feature selection. Swarm Evol Comput 2021;62:100847. https://doi.org/10.1016/j.swevo.2021.100847.

[18] Harary F. Graph Theory. Addison-Wesley Publishing Company; 1969.

[19] Hochba DS. Approximation algorithms for NP-hard problems. ACM SIGACT News 1997;28(2):40–52.

[20] Hoerl AE, Kennard RW. Ridge regression: applications to nonorthogonal problems. Technometrics 1970;12(1):69–82.

[21] Hsieh Y-A, Tsai YJ. Machine learning for crack detection: review and model performance comparison. J Comput Civ Eng 2020;34(5):04020038. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000918.

[22] Hussain K, Neggaz N, Zhu W, Houssein EH. An efficient hybrid sine-cosine Harris hawks optimization for low and high-dimensional feature selection. Expert Syst Appl 2021;176:114778.

[23] Johnson EA, Lam H-F, Katafygiotis LS, Beck JL. Phase I IASC-ASCE structural health monitoring benchmark problem using simulated data. J Eng Mech 2004;130(1):3–15.

[24] Karaboga D. An idea based on honey bee swarm for numerical optimization. Citeseer; 2005.

[25] Kaveh A. Advances in Metaheuristic Algorithms for Optimal Design of Structures. 3rd edition. Switzerland: Springer International Publishing; 2021.

[26] Kaveh A. Structural Mechanics: Graph and Matrix Methods, Research Studies Press. 3rd edition,. Hertfordshire, UK: Baldock; 2004.

[27] Kaveh A, Amirsoleimani P, Dadras Eslamlou A, Rahmani P. Frequency-constrained optimization of large-scale dome-shaped trusses using chaotic water strider algorithm. Structures 2021;32:1604–18. https://doi.org/10.1016/j.istruc.2021.03.033.
[28] Kaveh A, Eskandari A, Movasat M. Buckling resistance prediction of high strength steel columns using metaheuristic-trained artificial neural networks. Structures 2023;56(C):104853.
[29] Kaveh A, Eslamlou AD. Metaheuristic Optimization Algorithms in Civil Engineering: New Applications. Springer; 2020.
[30] Kaveh A, Eslamlou AD. Water strider algorithm: a new metaheuristic and applications. In: Structures. Elsevier; 2020. p. 520–41.
[31] Kaveh A, Nikbakht M. Block diagonalization of Laplacian matrices of symmetric graphs via group theory. Int J Numer Meth Eng 2007;69(5):908–47.
[32] Kaveh A, Rahmani P, Dadras Eslamlou A. Guided water strider algorithm for structural damage detection using incomplete modal data. Iranian. J Sci Technol 2022;46(2):771–88.
[33] Kennedy J, Eberhart R Particle swarm optimization. In: Proceedings of ICNN'95-international conference on neural networks, 1995. IEEE, pp 1942-1948.
[34] Kılıç F, Kaya Y, Yildirim S. A novel multi population based particle swarm optimization for feature selection. Knowl-Based Syst 2021;219:106894. https://doi.org/10.1016/j.knosys.2021.106894.
[35] Klein LA. Sensor and data fusion: a tool for information assessment and decision making, vol 138. SPIE Press; 2004.
[36] Kruskal WH, Wallis WA. Use of ranks in one-criterion variance analysis. J Am Stat Assoc 1952;47(260):583–621.
[37] Lin J-C-W, Djenouri Y, Srivastava G. Efficient closed high-utility pattern fusion model in large-scale databases. Information Fusion 2021;76:122–32. https://doi.org/10.1016/j.inffus.2021.05.011.
[38] Lin J-W, Djenouri Y, Srivastava G, Fourier-Viger P. Efficient evolutionary computation model of closed high-utility itemset mining. Appl Intell 2022;52(9):10604–16.
[39] Liu H, Setiono R Chi2: Feature selection and discretization of numeric attributes. In: Proceedings of 7th IEEE International Conference on Tools with Artificial Intelligence, 1995. IEEE, pp 388-391.
[40] Liu Y, Mu Y, Chen K, Li Y, Guo J. Daily activity feature selection in smart homes based on pearson correlation coefficient. Neural Process Lett 2020;51(2):1771–87.
[41] Mafarja M, Aljarah I, Faris H, Hammouri AI, Al-Zoubi AM, Mirjalili S. Binary grasshopper optimisation algorithm approaches for feature selection problems. Expert Syst Appl 2019;117:267–86.
[42] Mirjalili S, Lewis A. S-shaped versus V-shaped transfer functions for binary particle swarm optimization. Swarm Evol Comput 2013;9:1–14.
[43] Mosleh A, Montenegro P, Alves Costa P, Calçada R. An approach for wheel flat detection of railway train wheels using envelope spectrum analysis. Structure and Infrastructure Engineering:1–20 2020.
[44] Mosleh A, Montenegro PA, Costa PA, Calçada R. Railway Vehicle Wheel Flat Detection with Multiple Records Using Spectral Kurtosis Analysis. Appl Sci 2021;11(9):4002.
[45] Nguyen BH, Xue B, Zhang M. A survey on swarm intelligence approaches to feature selection in data mining. Swarm Evol Comput 2020;54:100663.
[46] Ouadfel S, Abd Elaziz M. Enhanced Crow Search Algorithm for Feature Selection. Expert Syst Appl 2020;159:113572. https://doi.org/10.1016/j.eswa.2020.113572.
[47] Pan H, Azimi M, Gui G, Yan F, Lin Z. Vibration-Based Support Vector Machine for Structural Health Monitoring. In: International Conference on Experimental Vibration Analysis for Civil Engineering Structures. Springer; 2017. p. 167–78.
[48] Pan H, Azimi M, Yan F, Lin Z. Time-Frequency-Based Data-Driven Structural Diagnosis and Damage Detection for Cable-Stayed Bridges. J Bridg Eng 2018;23(6).
[49] Quinlan JR. C4. 5: programs for machine learning. Elsevier; 2014.
[50] Rao H, Shi X, Rodrigue AK, Feng J, Xia Y, Elhoseny M, et al. Feature selection based on artificial bee colony and gradient boosting decision tree. Appl Soft Comput 2019;74:634–42.
[51] Santosa F, Symes WW. Linear inversion of band-limited reflection seismograms. SIAM J Sci Stat Comput 1986;7(4):1307–30.
[52] Santucci V, Baioletti M, Milani A. An algebraic framework for swarm and evolutionary algorithms in combinatorial optimization. Swarm Evol Comput 2020;55:100673. https://doi.org/10.1016/j.swevo.2020.100673.
[53] Shafer G. A mathematical theory of evidence,, vol 42. Princeton University Press; 1976.
[54] Shunmugapriya P, Kanmani S. A hybrid algorithm using ant and bee colony optimization for feature selection and classification (AC-ABC Hybrid). Swarm Evol Comput 2017;36:27–36.
[55] Silva MF, Santos A, Santos R, Figueiredo E, Costa JCWA. Damage-sensitive feature extraction with stacked autoencoders for unsupervised damage detection. Struct Control Health Monit 2021;28(5):e2714.
[56] Song Q, Jiang H, Liu J. Feature selection based on FDA and F-score for multi-class classification. Expert Syst Appl 2017;81:22–7. https://doi.org/10.1016/j.eswa.2017.02.049.
[57] Wolpert DH, Macready WG. No free lunch theorems for optimization. IEEE Trans Evol Comput 1997;1(1):67–82.
[58] Xu Y, Huang H, Heidari AA, Gui W, Ye X, Chen Y, et al. MFeature: Towards high performance evolutionary tools for feature selection. Expert Syst Appl 2021;186:115655. https://doi.org/10.1016/j.eswa.2021.115655.
[59] Yager RR. On the Dempster-Shafer framework and new combination rules. Inf Sci 1987;41(2):93–137.
[60] Yager RR. Dempster-Shafer belief structures with interval valued focal weights. Int J Intell Syst 2001;16(4):497–512.
[61] Yang X-S. Firefly algorithms for multimodal optimization. In: International symposium on stochastic algorithms. Springer; 2009. p. 169–78.
[62] Ying Y, Garrett JH, Oppenheim IJ, Soibelman L, Harley JB, Shi J, et al. Toward Data-Driven Structural Health Monitoring: Application of Machine Learning and Signal Processing to Damage Detection. J Comput Civ Eng 2013;27(6):667–80. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000258.
[63] Zadeh LA. Review of a mathematical theory of evidence. AI Mag 1984;5(3):81.
[64] Zhao Z, Morstatter F, Sharma S, Alelyani S, Anand A, Liu H (2010) Advancing feature selection research. ASU feature selection repository:1-28.
[65] Zhou Y, Kang J, Kwong S, Wang X, Zhang Q. An evolutionary multi-objective optimization framework of discretization-based feature selection for classification. Swarm Evol Comput 2021;60:100770. https://doi.org/10.1016/j.swevo.2020.100770.