Applied Soft Computing Journal 151 (2024) 111141

Contents lists available at ScienceDirect

Applied Soft Computing

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

Improved binary differential evolution with dimensionality reduction mechanism and binary stochastic search for feature selection

Behrouz Ahadzadeh[a], Moloud Abdar[b,*], Fatemeh Safara[c], Leyla Aghaei[d], Seyedali Mirjalili[e,f], Abbas Khosravi[b], Salvador García[g], Fakhri Karray[h,i], U.Rajendra Acharya[j]

[a] Department of Electrical, Computer and IT Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran
[b] Institute for Intelligent Systems Research and Innovation (IISRI), Deakin University, Geelong, Australia
[c] Department of Computer Engineering, Islamshahr Branch, Islamic Azad University, Islamshahr, Iran
[d] Department of Biomedical Engineering, Khomeinishahr Branch, Islamic Azad University, Isfahan, Iran
[e] Centre of Artificial Intelligence Research and Optimisation, Torrens University Australia, Brisbane, Australia
[f] University Research and Innovation Center, Obuda University, 1034 Budapest, Hungary
[g] Andalusian Institute of Data Science and Computational Intelligence, Department of Computer Science and Artificial Intelligence, University of Granada, Spain
[h] Centre for Pattern Analysis and Machine Intelligence, Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada
[i] Department of Machine Learning, Mohamed bin Zayed University of Artificial Intelligence, Abu Dhabi, United Arab Emirates
[j] School of Mathematics, Physics and Computing, University of Southern Queensland, Springfield, Australia

HIGHLIGHTS

- Proposing an optimal feature selection algorithm for medical datasets.
- Applying binary differential evolution approach and local search algorithm.
- Applying dimensionality Removal mechanism.
- Validating on heart disease, several cancer and RNA-seq COVID-19 datasets.
- Obtaining outstanding performance in generating optimal feature subset.

ARTICLE INFO

Keywords:
Feature selection
Evolutionary computation
Differential evolution
Binary stochastic search
Dimensionality reduction
Gene selection
Optimization

ABSTRACT

Computer systems store massive amounts of data with numerous features, leading to the need to extract the most important features for better classification in a wide variety of applications. Poor performance of various machine learning algorithms may be caused by unimportant features that increase the time and memory required to build a classifier. Feature selection (FS) is one of the efficient approaches to reducing the unimportant features. This paper, therefore, presents a new FS, named BDE-BSS-DR, that utilizes Binary Differential Evolution (BDE), Binary Stochastic Search (BSS) algorithm, and Dimensionality Reduction (DR) mechanism. The BSS algorithm increases the search capability of the BDE by escaping from local optimal points and exploring the search space. The DR mechanism then reduces the dimensions of the search space gradually. As a result of using DR, the local optima of the search space and the problem of wrong removal of important features before starting the search

Abbreviations: FS, Feature Selection; DE, Differential Evolution; BDE, Binary Differential Evolution; DR, Dimensionality Reduction; BSS, Binary Stochastic Search; EC, Evolutionary Computation; PSO, Particle Swarm Optimization; SFE, Simple, Fast, and Efficient; MTPSO, Multitasking PSO; BIBE, Binary Individual Search Strategy-based Bi-Objective Evolutionary; MFI-RFPA, Multifilter Integration and a Recursive Flower Pollination Algorithm; HFSIA, Hybrid Feature Selection Method based on Artificial Immune Algorithm; SIFE, Set-based Integer-Coded Fuzzy Granular Evolutionary; MF-CSO, Multiple Filtering Competitive Swarm Optimizer; FS-DOS, Dynamic Operator Selection Strategy; SaWDE, Self-adaptive Weighted DE; MDEFS, Modified DE approach to FS; GWO, Grey Wolf Optimization; MOFS, Multi-Objective Feature Selection; VGS-MOEA, Variable Granularity Search-based Multiobjective Evolutionary Algorithm; ESAPSO, Explicit and Size-Adaptive PSO; DimReM, Dimensionality Reduction Mechanism; SM-MOEA, Steering-Matrix-based Multiobjective EA; ABC, Artificial Bee Colony; MGWO, Modified Grey Wolf Optimization; PCC, Pearson Correlation Coefficient; IGR, Information Gain Ratio; TV, Term Variance; Std, Standard Deviation; CA, Classification Accuracy; SF, Selected Features; AVG, Average; EA, Evolutionary Algorithm; KNN, K-Nearest Neighbors; SVM, Support Vector Machine; ML, Machine Learning; SU, Symmetric Uncertainty.

* Corresponding author.
E-mail address: m.abdar1987@gmail.com (M. Abdar).

https://doi.org/10.1016/j.asoc.2023.111141
Received 25 April 2023; Received in revised form 28 November 2023; Accepted 5 December 2023
Available online 13 December 2023

process are reduced. The algorithm's efficiency is evaluated on 20 different medical datasets. The obtained outcomes indicate that the BDE-BSS-DR outperforms the BDE and BDE-BSS algorithms significantly. Furthermore, the effectiveness of the proposed algorithms in selecting the most important features of the heart disease data, several cancer diseases, and COVID-19 are also compared with several other state-of-the-art methods. Our results show that the BDE-BSS-DR with SVM classifier has a significant advantage over other methods with an average classification accuracy of 95.05% in heart disease and 99.40% in COVID-19 disease. In addition, the comparisons made with KNN and SVM classification prove the efficiency of the DR and BSS in generating a subset of optimal and informative features.

1. Introduction

Efficiency improvement in various data mining and ML applications is a benefit of dimensionality reduction, which is a critical step in data pre-processing. The curse of dimensionality is an issue that learner models often encounter when dealing with a considerable number of features, in particular in high-dimensional datasets [1]. In addition, the large number of features means more computational resources and memory usage in ML and data mining applications. These problems make ML methods inefficient and may reduce their scalability. Therefore, dimensionality reduction methods can be used as an effective method to reduce the above issues. Two types of methods widely used for reducing the number of features in a dataset are feature selection and feature extraction methods [2].

The purpose of feature extraction algorithms is to reduce the dimensionality of data by generating new features from existing ones. In contrast, FS algorithms identify an optimal subset of features and remove redundant, irrelevant, trivial, and noisy features to improve the speed and accuracy of ML methods [3]. The curse of dimensionality, as well as memory consumption and high computational costs, can be mitigated by using effective feature extraction and FS algorithms [3]. In recent years, feature extraction and FS algorithms have attracted researchers' attention, particularly in medical applications. Selecting a proper subset of the most significant features can determine the cause of diseases in the early stages and reduce medical care costs [4]. For instance, extracting information from medical images could help specialists diagnose a disorder. It could also help the specialists reach a consensus on some patients' medical conditions [4]. The selection of the most important features from the medical dataset makes it possible to discover the most significant and influential factors in various diseases and thus prevent various diseases. FS can also play a meaningful role in designing medical decision support systems more efficiently.

The process of FS involves the selection of a relevant subset of features while eliminating unimportant features from the original feature set [3]. One of the most effective ways to choose a feature subset is by utilizing search methods. The exhaustive search method involves searching the entire problem search space and examining all possible feature subsets. Feature selection is an NP-hard problem, and in a dataset with n features, the search space has $2^n - 1$ solutions [3,4,5]. However, with the exponential expansion of the number of proposed solutions to the problem [5], it is practically impossible to implement an exhaustive search algorithm in datasets with numerous features, even with the most advanced computers available.

Recently, a plethora of EC methods have emerged for FS. These techniques are recognized for their global search ability and reasonable time and memory complexity in FS [3]. However, despite their effectiveness in low-dimensional data, the huge search space and local optimal points in high-dimensional data often lead to poor performance. Therefore, it is imperative to implement strategies that enhance the efficiency of EC methods and maximize FS effectiveness, particularly in high-dimensional datasets. Various ideas have been presented by researchers to solve the challenges mentioned above. In this section we will examine two categories:

1. Using strategies to reduce the dimensions of the extremely large search space of high-dimensional data and using EC methods to search for the optimal subset in the reduced search space. Among these techniques, we can refer to SFE-PSO [3], BIBE [6], MFI-RFPA [7], MTPSO [8] and HFSIA [9]. For example, in [8], important features are determined using the ReliefF [10] algorithm, then less important features are filtered from the dataset, and finally PSO is used to search the optimal subset. Similarly, in [9], the Fisher method is used to determine the important features, and then the less important features are removed from the dataset. Then, the Artificial Immune Optimization algorithm is used to search for the optimal subset in the reduced search space. Although reducing the search space with filter-based algorithms is a good idea, using this idea still has problems. For instance, in most of these methods, before starting the search process a large number of features that the filter-based algorithm has recognized as less important features are removed from the dataset. The wrong removal of important and relevant features from the dataset causes the algorithm to not achieve the expected performance. Therefore, the efficiency of these methods is highly dependent on the filter-based algorithm used, and these methods may not have the expected efficiency in some datasets.
2. In another category of presented methods, the search process is performed in the original search space. Among these techniques, we can refer to SFE [3], SIFE [11], FS-DOS [12], SaWDE [13] and MDEFS [14]. For example, in [13] and [14], the DE algorithm with different mutation methods is used to select features in the original search space with high-dimensional data. Searching in the original search space eliminates the problem of mistakenly deleting important and relevant features. But these methods also have a problem. In the search process of the EC methods, making extensive changes in the solutions is not possible. Therefore, when these methods are used to search the extremely large search space of high-dimensional data with a large number of local optima, they encounter problems such as stopping at local optima, premature convergence, and high computational cost.

1.1. Objectives and contributions

The central objective of this research paper is to enhance the effectiveness of FS by introducing a novel DR mechanism and a novel BSS algorithm. The DR mechanism is proposed to reduce the problem of wrongly removing relevant and important features from the dataset. This method gradually removes less important features during the search process, unlike most of the methods presented in recent years that remove a large number of features before the start of the search process. The gradual removal of less important features causes the gradual decrease of search space dimensions and the problem of wrong removal of relevant features is reduced. Moreover, the BSS algorithm is proposed to reduce problems such as stopping at local optimal points and early convergence of FS, in particular in high-dimensional data. With a stochastic approach, BSS is proposed to increase the efficiency of the search and the escape of the trapping in local optimal points. The following are the main contributions of this research work:

- Introduce an efficient FS algorithm based on the DE algorithm, a novel DR mechanism, and a new BSS.
- Introduce a novel DR mechanism to reduce the problem of removing important and relevant features from the search space. In addition, the proposed DR transforms a large search space into a smaller one to increase search algorithms' efficiency in the FS process and solve the problem of trapping local optimal points in high-dimensional datasets.
- Propose a novel stochastic approach for enhancing the effectiveness of EC methods for FS through a BSS. The approach follows to goals: first, search for better solutions in some parts of the search space that the BDE algorithm does not have the ability to search, and second, prevent the algorithm from stopping at local optimal points.
- Application of the proposed algorithm in selecting proper features of the heart disease dataset.
- Application of the FS algorithm for gene selection of several cancer datasets and COVID-19 disease datasets.

The structure of this study is as follows: In Section 2, a comprehensive review of recent literature related to the research is presented and analyzed. Section 3 presents the proposed BDE-BSS and BDE-BSS-DR algorithms. In Section 4, the experimental outcomes conducted for performance evaluation are discussed. The paper concludes in Section 5, where the findings are summarized, and directions are suggested for future research.

2. Literature review

In the past few years, EC techniques have gained significant popularity due to their fast implementation, ability to find near-optimal solutions, and stochastic nature that allows them to escape from local optima. EC techniques such as Genetic Algorithm [15], DE [13], PSO [8], GWO [16], and other algorithms have been widely used for FS, achieving good performance. This section will provide an overview of some of the EC methods employed for wrapper-based and hybrid FS in recent research.

2.1. Wrapper-based FS algorithms

Zorarpacı et al. [17] introduced a new wrapper approach for FS that integrates the DE and ABC algorithms. The wrapper approach consists of three novel strategies: a binary neighborhood search, a modified onlooker bee method, and a binary mutation. The three strategies were devised to yield optimal feature subsets.

Mafarja et al. [18] introduced two new mechanisms to increase the performance of the grasshopper optimization method for FS. The first mechanism employs a pair of transformation functions to convert the continuous space into a binary space, whereas the second mechanism introduces a novel mutation operator to enhance the exploration ability of the algorithm.

El-Kenawy and colleagues [19] proposed an enhancement of the gray wolf algorithm by utilizing the stochastic fractal search algorithm in FS. The objective of this enhancement is to strike a balance between the capacity to explore and exploit, leading to an enhancement in effectiveness.

Zhang and team [20] introduced a FS algorithm named MOFS-BDE that uses a binary differential evolution approach with a self-learning strategy. To improve the DE efficiency in FS, three novel operators were employed. The binary mutation operator assists population members in swiftly locating optimal regions. The non-dominated sorting operator decreases the computational expense of the selection operator of the DE algorithm. The one-bit purifying search operator improves the self-learning ability of the elite population members situated in the optimal areas.

Tubishat and co-authors [21] improved the efficiency of the Salp swarm optimization by implementing Singer's chaotic map and a local search algorithm for FS. Singer's chaotic map method is used to enhance the diversity of population members. The local search also improves the exploitation capability of the algorithm, leading to the identification of a better solution. Tan et al. [22] suggested DimReM, a method that utilizes EC methods to reduce high-dimensionality search spaces. The primary objective of this method is to eliminate unimportant features to decrease the problem space. However, EC methods have inherent characteristics that may cause trapping best population members in a local optimum, which can result in trapping others in the population as well. Moreover, reducing the search space size to achieve the desired answers can be slow, which may lead to a high computational cost.

In their work, Wang et al. [13] presented a new method, SaWDE, for feature selection that utilizes a self-adaptive weighted DE algorithm. SaWDE incorporates a multi-population mechanism that helps to increase population diversity and identify optimal feature subsets. To cater to different datasets, the algorithm includes two self-adaptive mechanisms that aid in selecting appropriate mutation operators and parameter values. Additionally, a model is employed to assign weights to the features indicating the importance of each feature. However, the inclusion of self-adaptive and weight model mechanisms makes the algorithm computationally expensive.

In another study, Cheng et al. [23] proposed an FS algorithm for high-dimensional data called SM-MOEA, which is based on a steering-matrix and uses a multi-objective evolutionary approach. SM-MOEA aims to determine the importance of features using a steering-matrix and guide the population towards optimal solutions. To decrease the size of the search problem and enhance the performance of the algorithm, dimensionality reduction and individual repairing operators are utilized. The dimensionality reduction operator removes unimportant features, while the individual repairing operator repairs and maintains the search agents' good features. However, constructing a steering-matrix for datasets with numerous features can be memory-intensive. Moreover, incorrect feature removal during the dimensionality reduction phase may lead to local optimums and poor algorithm performance.

2.2. Hybrid FS algorithms

A novel FS algorithm was proposed by Song et al. [24] that utilizes particle swarm optimization and mutual information methods. The mutual information method determines the importance of each feature, which is then employed in the initialization of the PSO to accelerate the convergence rate of the FS algorithm. Additionally, two local search techniques are integrated to enhance the exploitation capability of the PSO.

On the other hand, Kilic et al. [25] introduced the multi-population particle swarm optimization algorithm to improve the exploration capability. This approach assigns a random initialization to each search agent based on the rank obtained by the ReliefF feature ranking algorithm, and each agent conducts its search process simultaneously.

Tarkhaneh and colleagues [14] suggested a differential evolution algorithm for FS named MDEFS. The algorithm introduces three innovative strategies, including two novel mutation strategies and a crossover operator to attain an acceptable equilibrium between exploring new possibilities and exploiting existing knowledge, one must carefully navigate and manage the two strategies. The ultimate goal is to generate effective feature subsets.

In their study, Song et al. [26] suggested a hybrid approach for FS in high-dimensional datasets. The introduced method involves using variable-sized cooperative co-evolutionary PSO. The first step is to identify the importance of each feature by employing the SU algorithm, which reduces the number of dimensions in the search space. After that, smaller subspaces are formed, and multi-swarm PSO is applied to explore each subspace. To ensure population diversity, some methods are adopted throughout the search process, such as eliminating duplicate particles and adding new ones. The final solution is obtained by

Table 1
A summary of the most recent FS methods presented with a search space dimensionality reduction approach.

No.	Reference	Method	Category, Classifier	No. of datasets, Range
1	BIBE (2022) [6]	Reducing the dimensions of the search space by Fisher score. Using Genetic Algorithm Operators to Search the reduced search space.	Hybrid, KNN	12, [2000,12600]
2	MFI-RFPA (2023) [7]	Reducing the dimensions of the search space of the FPA algorithm before starting the search process using the filtering of less important features determined by the three algorithms IGR, Relief-F, and Chi-squared.	Hybrid, C4.5, Naive Bayes	18, [2000,22283]
3	MF-CSO (2023) [27]	Reducing the dimensions of the search space of the CSO before starting the search process by converting the search space into several tasks and filtering the less important features determined by the three algorithms PCC, Relief-F, and TV. Searching with the CSO algorithm and transferring knowledge between tasks.	Hybrid, KNN	18, [2420,12600]
4	SFE-PSO (2023) [3]	Searching a large space of high-dimensional data with the SFE algorithm, finding important and relevant features in the initial search phase, removing unimportant features from the search space, and using the PSO algorithm to search the reduced search space.	Wrapper, KNN	40, [1024,24481]
5	MGWO (2023) [28]	Reducing the dimensions of the search space of the modified GWO algorithm before starting the search process by using the filtering of less important features determined by Copula entropy and Relief-F algorithms.	Hybrid, KNN	10, [2308,12600]
6	ESAPSO (2023) [29]	Reducing the dimensions of the search space of the PSO algorithm through explicit particle representation and filtering the less important features determined by the Maximum Information Coefficient method.	Hybrid, KNN	10, [2308,12533]
7	VGS-MOEA (2022) [30]	Reducing the dimensions of the search space by grouping features (Using PCC, SU, K-Means and other algorithms) and using one bit to display several grouped features. Using Genetic Algorithm Operators to Search the reduced search space.	Hybrid, KNN	12, [1024,22283]

merging solutions acquired from various sub-swarms using a crossover technique. However, this algorithm has some drawbacks, such as significant computational costs caused by distinct sub-swarms and potential local optimal point entrapment because of the insufficient interaction among important features across various sub-swarms.

On the other hand, Chen et al. [8] proposed the MTPSO algorithm as a feature selection method for data with large dimensions. MTPSO uses a multitasking approach and PSO divides the large search spaces into smaller ones using a ReliefF-based method. The smaller sub-tasks are then searched using a multi-population PSO. To improve efficiency, knowledge transfer is employed to exchange information about important features among the sub-populations.

Seven recent techniques of reducing the dimensions of the search space and using EC methods for searching in the reduced search space have been summarized in Table 1.

2.3. Research gaps

As previously mentioned, EC-based FS algorithms have shown promising results. However, they often encounter issues such as premature convergence and being trapped in local optima solutions [3], which is particularly problematic for high-dimensional datasets. Furthermore, these algorithms face other obstacles such as the curse of dimensionality, expensive computations, and high memory usage [31]. The FS algorithms discussed in earlier sections have improved the efficiency of EC methods by employing various operators, local search techniques, and search space reduction methods. Nevertheless, some of the algorithms reviewed, including [13] and [23], may require significant memory and computational resources for FS. Additionally, certain algorithms, such as [6] and [7], that conducted dimension removal through filter-based methods may lead to local optima solutions in some datasets. This occurs when relevant features are not accurately identified, resulting in incorrect filtering from the dataset and poor classification performance. To summarize, the challenges associated with current FS algorithms are due to their limitations in handling high-dimensional search spaces, resulting in possible local optima solutions, high computational and memory costs, and inadequate classification efficacy. The challenges, existing solutions, challenges of the methods presented in recent years, research gaps of feature selection from high-dimensional data, and our proposed solutions are summarized in Table 2.

Thus, we present a novel FS approach that utilizes a combination of the BDE algorithm, a BSS algorithm, and a DR mechanism to address the challenges mentioned earlier. The BSS algorithm amplifies the search capacity of the FS algorithm, whereas the DR mechanism gradually reduces the search space's size, reducing the algorithm's memory and computational demands. Moreover, it optimizes the search process, thus facilitating effective searching. It is crucial to note that the No-Free-Lunch (NFL) theorem [36] suggests that no EC-based FS algorithm can identify the optimal feature subset across all datasets. This theorem has inspired our work to develop a new FS algorithm.

3. Model description and assumptions

EC methods are highly efficient in selecting features in low-dimensional and medium-dimensional datasets. One of the challenges of FS algorithms in low and medium-dimensional datasets is the possibility of getting trapped in local optima. In high-dimensional datasets, EC methods face further difficulties due to the vast search space and numerous local optima. Consequently, EC methods often suffer from issues such as the curse of dimensionality, trapping at local optima, and premature convergence [3]. DE is a successful EC technique that has been utilized in several recent FS algorithms [37,38], [39]. It was first presented by Storn and Price [40] in 1995. The DE leverages three commonly used operators, namely mutation, crossover, and selection, to look for the optimal solution. To search the problem space, the DE algorithm employs the mutation operator [37]. The crossover operator has a crucial impact in leading the algorithm to the optimal solutions to the problem by exchanging information between search agents and population agents, creating new search agents [41]. Lastly, it uses the selection operator to guide the search space population members toward potential optimal solutions [42,43,44]. DE has demonstrated satisfactory performance in various problems that require different levels of exploration and exploitation. The algorithm has been enhanced using self-organizing parameters, local search algorithms, new mutation and crossover operators, among other techniques [45].

The simple structure, easy implementation, strong search ability, and high efficiency of the differential evolution algorithm have used this algorithm in various optimization problems [41]. The mutation operator has an important impact on the exploitation and exploration capabilities of the DE algorithm. Some mutation methods, such as $DE/rand/1$, have high exploration ability but weak exploitation ability. So, the convergence speed of the algorithm is slow when using them. In contrast, some mutation methods, such as $DE/Best/1$, have high exploitation ability but

Table 2
The challenges, existing solutions and their challenges, research gaps of feature selection from high-dimensional data and solutions of this article.

Challenges of feature selection from high-dimensional data	Existing solutions	Challenges of the existing solutions	Research gaps and solutions proposed in this article
Curse of dimensionality, high computational cost and memory consumption	Removing a large number of irrelevant features before starting the search process with different methods such as determining important features with filter-based methods, and searching the reduced search space with EC methods to determine the optimal subset.[6,7,8,27,28,29].	The wrong removal of important features and the incorrect determination of the filter point to remove features from the search space causing poor performance.	Research gap: Avoid deleting important features before starting the search process. The proposed solution of this article: Gradual removal of unimportant features in the search process with Dimensionality Reduction (DR) mechanism.
	Optimal subset search in the original search space.[3,11,13,14,32,33,34,35].	Algorithm stops at the local optimal points of the extremely large space of high-dimensional data, high computational cost, and high memory consumption.	Research gaps: Finding a solution to increase the ability of the algorithm to escape from the local optimal points and the ability to search for points close to the global optimal. The proposed solution of this article to increase the search capability: Binary Stochastic Search algorithm. The proposed solution of this article to reduce the computational cost and memory consumption: Dimensionality Reduction (DR) mechanism.

poor exploration ability. For this reason, when these mutation methods are used, the algorithm may stop at a local optimum, leading to premature convergence. Therefore, to achieve high performance in various problems, achieving a well-balanced distribution of capabilities for both exploitation and exploration is imperative for optimal performance [41].

We, therefore, exhibit both BSS and DE algorithms to propose a novel FS algorithm. In this algorithm, the mentioned two mutation methods are used in different search stages. Also, the BSS algorithm plays a vital role in achieving this goal. In addition, a new method called the DR mechanism has been presented to increase the efficiency of FS. This study aims to improve the exploration and exploitation capacity of the DE algorithm via the implementation of a DR mechanism and BSS algorithm. Table 3 provides a summary of the notations utilized in the proposed algorithms.

3.1. Solution methods

3.1.1. Encoding and initialization of search agents

Originally, the DE algorithm was designed to solve continuous optimization problems; thus, it cannot be used to resolve binary problems like FS directly. This calls for appropriate modifications to adapt the algorithm to the requirements of the problem under study. Therefore, a binary version of the algorithm is required. This paper uses a binary version of the differential evolution algorithm (BDE) developed by Wang et al. [46]. Due to the binary form of the FS, the population members of the BDE must be initialized as binary. Assuming that the number of features or the dimensions is equal to D, each problem agent in the DE algorithm will have a D component; each component represents one of the features. In Fig. 1, we see one of the problem agents in binary form. Fig. 1, shows the initialization of a search agent in binary form. As shown the Fig. 1, the values 1 and 0 indicate that a feature is selected or not selected, respectively. In this section, the population members of the algorithm are initialized randomly using binary values.

3.1.2. Global search based on the BDE algorithm

The BDE algorithm executes the global search or exploration phase throughout the search, where the selection, crossover, and mutation operators are employed. Within the mutation stage, a probability estimation operator is utilized to determine the values of each search agent component. In iteration $it+1$ of the algorithm, for each component j of the ith search agent, Eq. (1) represents the implementation of the probability estimation operator: the probability estimation operator is used as (1):

$$P(x_{i,j,it}) = 1\Big/\left(1+e^{-2\times b\times(MO-0.5)/(1+2\times F)}\right), \tag{1}$$

$$MO = x_{r1,j,it} + F.\left(x_{r2,j,it} - x_{r3,j,it}\right), \tag{2}$$

where $P(x_{i,j,it})$ is the probability estimation operator, which is calculated for each agent component separately, and b is the bandwidth factor, and it is a positive integer used to adjust the extent and outline of the probability estimation operator. The proper value of this variable can play an influential role in increasing search efficiency and keeping population diversity high. F represents the scale factor, and MO denotes the mutation operator of the original DE algorithm, which is calculated by the values of the three search agents of population members, and $x_{r1,j,it}, x_{r2,j,it}$ *and* $x_{r3,j,it}$ represent the jth component of the three search agents that are randomly selected. $r1$, $r2$, and $r3$ are three random numbers to select population agents [46]. After generating three random numbers in the proposed method, the MO value is determined as follows.

$if\ ImIter\ > Max_{ImIter\_BDE}$

Table 3
Notations used in the proposed algorithms.

Notations	Description
x_{best}	The best solution for the population
x_{new}	The solution obtained by the BSS algorithm
x_{worst}	The worst solution in the population
$ImIter$	Is a counter, and if the algorithm cannot improve the best solution in each iteration, a unit is added to its value
Max_{ImIter_BDE}	Maximum number of improvement iterations for switching between mutation methods of the BDE algorithm
BSS_{Max_it}	The maximum number of improvement iterations considered to call the BSS algorithm
Max_{ImIter_DRM}	The maximum number of improvement iterations considered to call the DR mechanism
N_{ch}	The number of changes in the best solution
Ch_r	Rate of change
N_S	The number of features after using the DR mechanism
C_t	Calling time of the BSS algorithm
R_t	Removal threshold
NS_r	Rate of non-selection of features

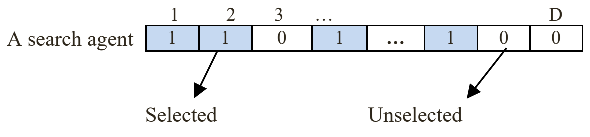


Fig. 1. Encoding of a search agent.

$$MO = x_{r1,j,it} + F.\left(x_{r2,j,it} - x_{r3,j,it}\right)$$

$$\begin{aligned}&else\\ &MO = x_{best,j,it} + F.\left(x_{r1,j,it} - x_{r2,j,it}\right)\\ &end\end{aligned} \tag{3}$$

where, *ImIter* is a counter, and if the algorithm cannot improve the best solution in each iteration, a unit is added to its value. Max_{ImIter_BDE} is the maximum number of improvement iterations for switching between mutation methods of the BDE algorithm. Also, x_{best} is the best solution in the current repetition *it*, which means its fitness is maximal so far. The reason for using x_{best} is sharing information obtained by the best agent among other search agents. For example, if the algorithm fails to improve the classification accuracy in Max_{ImIter_BDE} consecutive iterations, then the algorithm uses the $DE/rand/1$ mutation method to improve the exploration capability of the algorithm by generating random solutions. But if in Max_{ImIter_BDE} consecutive iterations, the algorithm can achieve better solutions. In that case, the $DE/best/1$ mutation method employs the information of the best solution obtained to direct the population to the best solution obtained. After that, the probability estimation operator value for each of the components is calculated using Eq. (4) using the following Equation:

$$v_{i,j,it} = \begin{cases} 1\ , & if \quad rand() \le P(x_{i,j,it}) \\ 0\ , & \text{Otherwise} \end{cases}. \tag{4}$$

In the above Equation, $v_{i,it}$ is the mutated vector of *i*th population member in iteration *it*. According to the above Equation, the components of the search agents, whose probability estimation value is closer to 1, have a better chance of becoming 1. The components with the smaller probability estimation value have a better chance of becoming 0. Therefore, the features selected by the majority of the agents are more likely to be selected, and the features with probabilities close to 0 are more likely to be removed. The basis of the work of the selection and crossover operators in the BDE is similar to the original DE algorithm. A brief explanation of the operators and the search process of the original DE algorithm can be found in Appendix 1.

3.1.3. *Binary Stochastic Search*

After the global search phase, which includes the use of the mutation, crossover, and selection operators, the BSS algorithm will be called to improve the x_{best} of the BDE. The pseudo-code of the BSS is presented in Algorithm 1. If the BDE algorithm fails to improve performance in $Max_{ImIterLSA}$ consecutive iterations, this algorithm will be executed. Therefore, a counter, *ImIter*, is used to count the consecutive repetitions that the algorithm could not find a better solution. If the algorithm cannot improve the best solution in each iteration, a unit is added to its value. The value of this counter is initialized with 0 after each iteration of the BSS algorithm.

Algorithm 1. Binary Stochastic Search (BSS).

```
Algorithm 1: Binary Stochastic Search (BSS)
$LS_{it} = 1$          % Number of BSS algorithm iterations
$x_{new} = x_{best}$
While $BSS_{Max\_it} \le BSS_{it}$ do
    $Ch_r =$ Select a number based on number of features in the dataset
    $N_{ch} = Ceil\ (Ch_r * $ number of features in the dataset$)$
    Select randomly n_ch features from $x_{new}$
    For each selected features in $x_{new}$
        If number of features in dataset $> 500$
            If $NS_r > Rand$
                If feature $== 1$
                    feature $= 0$
                End if
            Else                                  % Search process for high- dimensional
                If feature $== 0$                     dataset
                    feature $= 1$
                Else
                    feature $= 1$
                End if
            End if
        Else
            If feature $== 1$                     % Search process for medium and low-
                feature $= 0$                         dimensional dataset
            Else
                feature $= 1$
            End if
        End if
        If number of selected features in $x_{new} == 0$
            $x_{new} = x_{best}$
            Select randomly 1 feature from $x_{new}$
            feature $= 1$
        End if
    End for
    If $fit(x_{new}) \ge fit(x_{best})$
        $x_{best} = x_{new}$
    End if
    $BSS_{it} = BSS_{it} + 1$
End while
```

The goal of this algorithm is to improve the solution of the x_{best} of the population. Throughout the search algorithm, the features of the x_{best} of the population are changed to get a better solution. A number called the change rate (Ch_r) is first selected to conduct a BSS. Then, the value of the rate of change in different datasets is selected as follows:

$$Ch_r = \begin{cases} [0.02, 0.04, 0.06, 0.07, 0.08, 0.09, 0.1, 0.2] & D <= 100 \\ [0.005, 0.006, 0.007, 0.008, 0.01, 0.03, 0.05, 0.06, 0.08, 0.1, 0.2] & 100 < D < 500 \\ Randomly\ generated\ numbers\ between\ [0, 1] & D \ge 500 \end{cases} \tag{5}$$

where Ch_r is the rate of changes, and D is the number of features in the dataset. After determining the change rate, the number of changes is calculated using the following equation:

$$N_{ch} = \lceil Ch_r \times \quad D \rceil, \tag{6}$$

where N_{ch} refers to the count of changes involved. The N_{ch} parameter conveys the quantity of feature state transitions occurring during a single iteration of the BSS. For example, with a dataset containing 2000 features, if the number of changes is selected as 0.05, using Eq. (6), the number of changes would be 100. Therefore, the algorithm performs the stochastic search by making changes in 100 components of the $\boldsymbol{x}_{best}$. Now, 100 numbers are created in the range of 1 to the number of features in the dataset, randomly. In the following, two different approaches are used in the BSS algorithm based on the number of features in the dataset:

- **High-dimensional data:** In high-dimensional datasets such as the DNA microarray, many features are irrelevant. In such datasets, achieving the least number of features (genes) can demonstrate a significant role in identifying important factors in diseases and increasing the efficiency of ML methods. Therefore, to FS in these datasets, we need to change the number of more features to non-selection status (i.e., 0). To achieve this goal in the BSS algorithm, a parameter called non-selection rate (NS_r) is used. The value of this parameter indicates the chance of not selecting the features. Therefore, if we set values close to 1 for this parameter, the chance of not selecting features increases at each stage of the search. Therefore, a random number between [0,1] is generated first. If the value of this number is less than NS_r, then the value of all randomly selected features changes to 0. If the random number generated is greater than NS_r, if the value of randomly selected features is 0, it changes to 1, and if this value is 1, it changes to 0. In this way, we consider a larger probability of not selecting the features. This process causes important features to be selected without too many unimportant features.
- **Low-dimensional data:** In a low-dimensional dataset, especially a dataset of various diseases, because the features collected in the dataset are usually related to the disease, so about half the number of features can be important features. In these datasets, considering the equal probability of selecting and non-selecting features can improve the performance of the FS process. In this dataset, the value of all randomly selected features changes to 0 and 1 if they were 1 and 0, respectively.

After each iteration of the BSS algorithm, the objective function value of the $\boldsymbol{x}_{new}$ is compared to that of the $\boldsymbol{x}_{best}$. If the objective function value of $\boldsymbol{x}_{new}$ is greater than or equal to the $\boldsymbol{x}_{best}$, it is randomly replaced with the $\boldsymbol{x}_{best}$ or $\boldsymbol{x}_{worst}$. Replacing the obtained solution randomly with the $\boldsymbol{x}_{best}$ or $\boldsymbol{x}_{worst}$ prevents the algorithm from rapidly converging to the local optimums. The pseudo-code of the BDE-BSS algorithm is presented in Algorithm 2. First, the initialization of the search agents is done, and the algorithm enters the search process. In each iteration of the algorithm, first, the global search process is performed using the BDE algorithm, and then if the BSS algorithm is called, the local search is conducted using the search around the $\boldsymbol{x}_{best}$.

Algorithm 2. BDE-BSS Algorithm.

```
Algorithm 2: BDE-BSS Algorithm
Population initialization $pop = (x_{1,it}, x_{2,it}, \dots, x_{i,it}, x_{NP,it}), i = 1,2,\dots,NP$
Evaluate the fitness: $fit(x_{1,0}), \dots, fit(x_{NP,0})$
While $it \le Max_{it}$ do
    For $i = 1$ to $NP$ do
        For $i = 1$ to No. of features do
            If $it > 10$ and $ImIter > Max_{ImIter\_BDE}$
                $MO = x_{r1,j,it} + F.(x_{r2,j,it} - x_{r3,j,it})$
            Else
                $MO = x_{best,j,it} + F.(x_{r1,j,it} - x_{r2,j,it})$
            End if
            $v_{i,j,it} = \begin{cases} 1 & , \ if\ rand() \le P(x_{i,j,it}) \\ 0 & , \ \text{Otherwise} \end{cases}$
        End for
        Crossover Step
        Selection Step
        Evaluate the fitness: $fit(x_{1,0}), \dots, fit(x_{NP,0})$
    End for
    If $ImIter > Max_{ImIter\_LSA}$
        $x_{new}$ = Call Algorithm 1 to Improve $\boldsymbol{x}_{best}$
        If $x_{new} > \boldsymbol{x}_{best}$
            If $rand > 0.5$
                $\boldsymbol{x}_{best} = \boldsymbol{x}_{new}$
            Else
                $\boldsymbol{x}_{worst} = \boldsymbol{x}_{new}$
            End if
        End if
    End if
    $it = it + 1$
End while
```

3.1.4. Dimensionality reduction mechanism

In high-dimensional datasets, a significant portion of the features may lack importance, and therefore, to optimize the FS performance, a BSS algorithm was employed to eliminate redundant features. However, the sizeable search space coupled with the presence of multiple local optima may cause the BDE-BSS algorithm to converge at suboptimal points. As a solution to this challenge, an alternative algorithm named BDE-BSS-DR is proposed. BDE-BSS-DR reduces the size of the problem space to a more manageable size and therefore improves the performance. The pseudo-code of the BDE-BSS-DR algorithm is presented in Algorithm 3. In the BDE-BSS-DR algorithm, the FS process starts with Algorithm 2. After performing the search process, if BDE-BSS-DR cannot find better solutions in several consecutive iterations, it means that it has stopped at a local optimal point. In such cases, we use the dimensionality reduction mechanism. That's why this algorithm is called BDE-BSS-DR. DR mechanism consists of three steps: 1) Determining important and less important features, 2) Removing less important features and search agents, and 3) Re-initialization and re-evolution of search agents and continuing the search in the new search space.

Algorithm 3. BDE-BSS-DR Algorithm.

```
Algorithm 3: BDE-BSS-DR Algorithm
Population initialization: $pop = (x_{1,it}, x_{2,it}, \dots, x_{i,it}, x_{NP,it}), i = 1,2,\dots,NP$
Evaluate the fitness: $fit(x_{1,0}), \dots, fit(x_{NP,0})$
While $it \le Max_{it}$ do
    Use Algorithm 2 for feature selection
    If $it > C_t$ and $ImIter > Max_{ImIter\_DRM}$
        Find selected features in $\boldsymbol{x}_{best}$
        Find selected features by the most search agents
        Find top-ranked features by ReliefF algorithm
        $N_S$=Calculate the number of features after the dimensionality reduction step
        If $N_S > R_t$
            Deleting non-selected features by $x_{best}$, most individuals, and ...
            reliefF algorithm from the dataset and population
            Reinitialization and reevolution of the population in the new subspace
        End if
    End if
    $it = it + 1$
End while
```

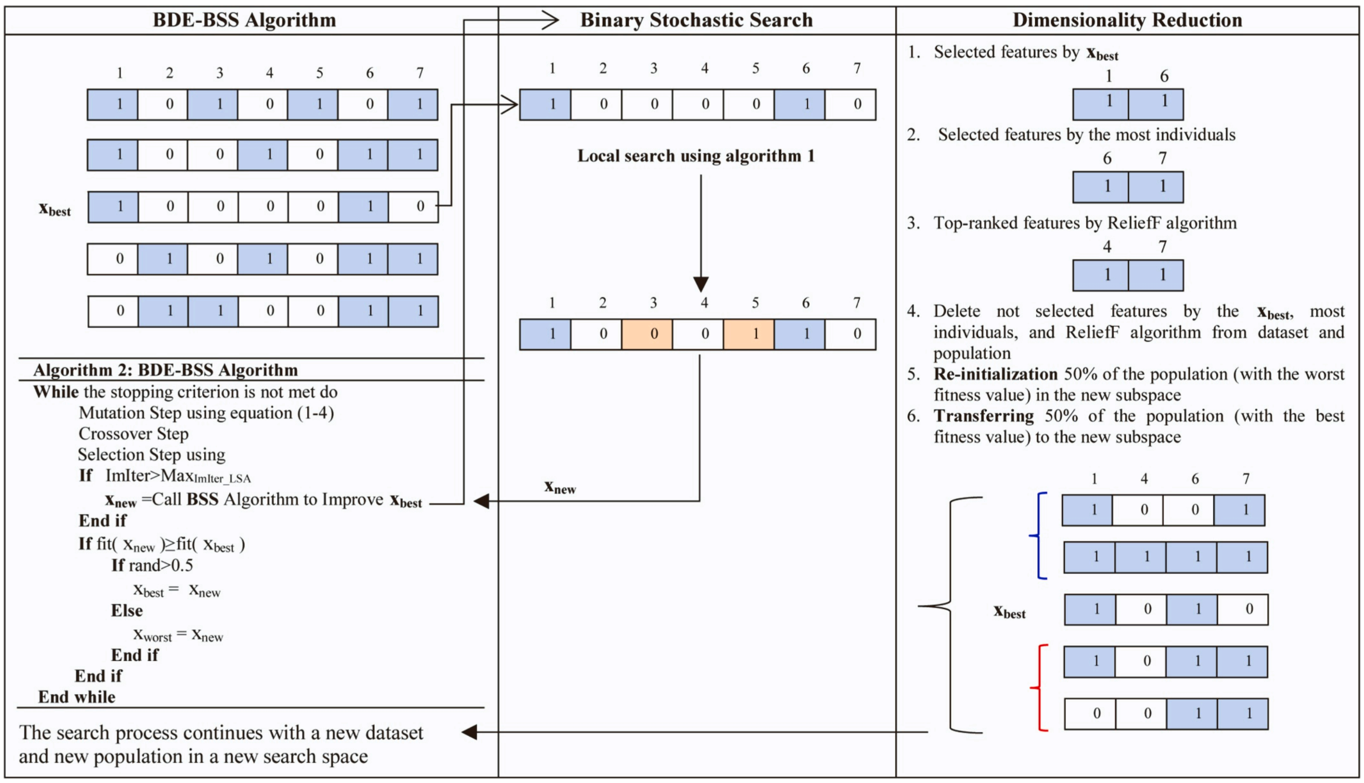
The first step to reduce the size of the problem space via the DR mechanism is to find important and unimportant features through the x_{best} of the population, the rest of the search agents, and the ReliefF algorithm. This step determines the important and unimportant features using the following three methods.

1. **Selected features by the x_{best}:** selected features by the x_{best} are important features, so these features should not be removed from the dataset during the physical removal step.
2. **Selected features by most search agents:** selected features by most search agents are important features. On the other hand, features that most search agents do not select are likely to be unimportant features. Therefore, identifying and removing unimportant features that most search agents have left non-selected can effectively reduce the search space. In this method, features that are not selected by more than half of the search agents are defined as unimportant features. Therefore, these features are good candidates for physical removal. To gradually reduce the size of the problem, the process of identifying important features begins with half of the population. For example, if the number of members of a population is 20, features 10 population agents select are identified as important features. In the later search stages, an agent is added to the number of population members involved in selecting important features each time the dimension reduction method is called. For example, in the second call of the DR mechanism, the number of search agents increases from 10 search agents to 11 search agents. This method makes the search space smaller as the final iterations of the algorithm approach
3. **Important features determined by ReliefF algorithm:** Using a filter-based algorithm can play an important role in determining important and unimportant features. Therefore, determining important features by the ReliefF algorithm and not physically removing them can play an effective role in deciding unimportant features. At each step of the DR mechanism call, 0.1 dataset features are used as the most important features defined by ReliefF. For example, in a dataset with 2000 features, 200 important features determined by ReliefF are used in the first call of the DR mechanism. In the second call, if the number of dataset features is reduced to 600 features through the DR mechanism, 60 important features are used.

Once the significant and insignificant features have been identified, the latter are eliminated from the dataset and search agents using a physical removal process. It is important to note that if the number of remaining features in the revised search space exceeds the designated threshold value (R_t) after the removal process, the deletion operation will be executed. In other words, if the delete operation causes the number of features to reach less than R_t, the deletion will not take place and the algorithm will continue the search process in the same search space as before. The parameter R_t is a remove threshold to use the DR mechanism. Therefore, the deletion process is carried out until the number of remaining features in the dataset is greater than the parameter R_t. Physical removal of features from the dataset causes the size of the problem space to be reduced compared to the dimensions of the original dataset. Therefore, Algorithm 2 can find better subsets under the new spaces. In addition to physically removing features from the dataset, unimportant features are also removed from the search agents. This way, the search space is reduced from a larger space to a smaller one.

To pursue the exploration process in the new space, the solution obtained by the x_{best} and half of the population agents are transferred to the new space. In other words, after the physical removal of unimportant features, the values of the other features are preserved in the x_{best} and half of the search agents (with the best fitness value). This process preserves the information obtained in the previous phases of the search to continue the search process. In addition, the other half of the population agents (with the worst fitness value) are randomly initialized in the new search space. Using this method increases the diversity of the population. Therefore, a new population is formed under the new reduced space, and the search process continues.

The DR mechanism removes unimportant features from the problem search space. Therefore, using Algorithm 2 can obtain a subset of


BDE-BSS Algorithm
Binary Stochastic Search
Dimensionality Reduction
$\mathbf{x_{best}}$
Local search using algorithm 1
$\mathbf{x_{new}}$
1. Selected features by $\mathbf{x_{best}}$
2. Selected features by the most individuals
3. Top-ranked features by ReliefF algorithm
4. Delete not selected features by the $\mathbf{x_{best}}$, most individuals, and ReliefF algorithm from dataset and population
5. **Re-initialization** 50% of the population (with the worst fitness value) in the new subspace
6. **Transferring** 50% of the population (with the best fitness value) to the new subspace
The search process continues with a new dataset and new population in a new search space

```
Algorithm 2: BDE-BSS Algorithm
While the stopping criterion is not met do
    Mutation Step using equation (1-4)
    Crossover Step
    Selection Step using
    If ImIter>Max_{ImIter_LSA}
        x_new =Call BSS Algorithm to Improve x_best
    End if
    If fit( x_new )≥fit( x_best )
        If rand>0.5
            x_best = x_new
        Else
            x_worst = x_new
        End if
    End if
End while
```

Fig. 2. Illustration of BDE-BSS-DR scheme.

optimal features. Fig. 2 shows the proposed BDE-BSS-DR algorithm in global search, BSS, and DR mechanism. As shown in Fig. 2., the search process starts using the BDE algorithm. Then, the BSS algorithm is called if the BDE algorithm cannot improve the search process. Finally, after several iterations of the BDE-BSS search algorithm, if there is no improvement in the search process, the DR mechanism is called to reduce the search space size. It is important to note that the DR mechanism may be called several times during the BDE-BSS-DR algorithm search process. In addition, determining the calling time of the DR mechanism can be optimal in increasing the efficiency of the FS algorithm. For example, in the proposed Algorithm 3, as we see in the pseudo-code, the DR mechanism is called if the number of iterations of the algorithm is more than C_t and the algorithm cannot improve performance in the previous Max_{ImIter_DRM} iterations.

3.1.5. Evaluation function

Every element in the population is a potential solution to the FS problem. To evaluate each of the solutions, the classification accuracy is defined as presented in Eq. (7):

$$fitness = \frac{T_P + T_N}{T_P + T_N + F_P + F_N} \times 100\%. \tag{7}$$

For medical dataset classification, *TP* and *TN* are the numbers of samples of sick and healthy people that the classifier has correctly classified. *FP* indicates the number of sick people that the classifier has wrongly classified as healthy, and *FN* indicates the number of healthy people that the classifier has wrongly classified as sick. The confusion matrix and efficiency criteria for the binary classification can be found in Appendix 2.

3.1.6. Analysis of computational complexity

The BDE algorithm's computational complexity can be expressed as $O(Max_{it}(D\times\ NP+\ C\times\ NP))$, where Max_{it} denotes the maximum number of iterations. D represents the dataset dimensions or the number of features, while C signifies the cost of the fitness function evaluation, and NP denotes the dimensions of the population. The BSS algorithm's computational complexity is $O(Max\_BSS_{it}(D+\ C))$, where BSS_{Max_it} represents the maximum number of repetitions. In the worst case, the BSS algorithm is executed after every five iterations of the BDE algorithm, $O(\frac{Max_{it}}{Max_{ImIter\_BDE}}\times\ BSS_{Max\_it}(D+\ C))$. Therefore, the computational complexity of BDE-BSS in the worst case is $O\Big(Max_{it}\quad (D\times NP + C\times NP) + \frac{Max_{it}}{Max_{ImIter\_BDE}}\times BSS_{Max\_it}(D+C)\Big)$.

The computational complexity of the ReliefF algorithm is $O(N^2\times D)$, where N denotes the number of samples in the dataset. In the worst case, the DR mechanism is called after C_tth iteration and after every Max_{ImIter_DRM} iterations of the BDE-BSS algorithm, $O(\frac{Max_{it}-C_t}{Max_{ImIter\_DRM}}\times N^2\times D)$. Therefore, the computational complexity of BD-BSS-DR in the worst case is $O\Big(Max_{it}\quad (D\times NP + C\times NP) + \frac{Max_{it}}{Max_{ImIter\_BDE}}\times BSS_{Max\_it}(D+C) + \frac{Max_{it}-C_t}{Max_{ImIter\_DRM}}\times N^2\times D\Big)$.

Using the BSS algorithm in high-dimensional datasets with huge search space and finding solutions with fewer features reduces the cost of calculating the evaluation function. Furthermore, the use of the DR mechanism in high-dimensional data is crucial in reducing the computational cost of the algorithm during the search process. By transforming the large search space into a smaller one, the algorithm becomes more efficient in identifying optimal feature subsets.

4. Experiments and discussions

4.1. Benchmark datasets

This section evaluates the efficiency of the proposed algorithms on 20 medical datasets. The website addresses for downloading the used datasets can be found in Appendix 3. The general specification of the datasets is presented in Table 4. Datasets 1 to 3 are low-dimensional, 4 to 8 are medium-dimensional, and the rest are high-dimensional. The reason for choosing these datasets is the number of their various features, which allows us to evaluate the proposed algorithms with different characteristics.

4.2. Experiment design

The paper's experiments were conducted using Matlab R2019a on a system with an Intel (R) core (TM) i5–5200 U CPU, 2.2 GHz, and 12 gigabytes of RAM. The experiments and evaluations are performed in two parts: 1) comparison of the proposed algorithms with the BDE algorithm and presented DE-based FS algorithms in recent years, and 2) Application of the proposed BDE-BSS and BDE-BSS-DR algorithms in FS of coronary artery disease dataset and gene selection of COVID-19 and several cancer datasets and comparing the results with the methods presented in recent years. Each experiment sets different parameters for the proposed and other algorithms. Therefore, the parameters used in the tests of each section are mentioned.

4.3. Results and analysis

4.3.1. Comparison of the proposed algorithms and DE-based FS algorithms

This section aims to evaluate and compare the effectiveness of the

Table 4
Description of experimental medical datasets.

No.	Dataset	No. of features	No. of samples	No. of classes	Categories
1	Dermatology	34	366	6	low
2	SPECTF heart	44	267	2	low
3	Z-Alizadeh Sani	54	303	2	low
4	Scadi	206	70	7	medium
5	Arrhythmia	279	452	16	medium
6	LSVT	310	126	2	medium
7	Pomeroy-2002-v1	857	34	2	medium
8	Alizadeh-2000-v1	1095	42	2	medium
9	Colon	2000	62	2	high
10	SRBCT	2308	83	4	high
11	Leukemia	5327	72	3	high
12	Bladder Cancer	5724	40	3	high
13	DLBCL	7129	77	2	high
14	CNS	7129	60	2	high
15	ALL-AML	7129	72	2	high
16	Brain_Tumor2	10367	50	4	high
17	Prostate Tumor	10509	102	2	high
18	MLL	12533	72	3	high
19	Breast Cancer	24481	97	2	high
20	COVID-19	25534	90	3	high

Table 5
Parameter setting for the applied algorithms.

Algorithm	Parameter values
BDE [46]	$CR = 0.2, F = 0.8, NP = 30, b = 20$
DimReM-BDE [22]	$CR = 0.2, F = 0.8, NP = 30, b = 20$
SaWDE [13]	$CR = [0.1, 0.2, 0.9, 0.8, 0.9, 0.1, 0.8, 0.2]$, $F = [0.5, 1, 0.6, 0.9, 0.5, 0.9, 0.6, 1], NP = 100$
BDE-BSS	$F = 0.8$, $CR = 0.2$, $b = 20$, $NP = 21$, $BSS_{Max\_it} = 50$, $NS_r = 0.8$, $Max_{ImIter\_BDE} = 5$, $Max_{ImIter\_BSS} = 5$
BDE-BSS-DR,	$F = 0.8$, $CR = 0.2$, $b = 20$, $NP = 21$, $BSS_{Max\_it} = 50$, $NS_r = 0.8$, $Max_{ImIter\_BDE} = 5$, $Max_{ImIter\_BSS} = 5$,
BDE-BSS-DR-r	$C_t = \begin{cases} 50D < 2000 \\ 100D \geq 2000 \end{cases}, \quad Max_{ImIter_{DRM}} = \begin{cases} 10D < 2000 \\ 20D \geq 2000 \end{cases}, \quad R_t = 30$

newly introduced algorithms with the existing ones including the BDE [46], DimReM-BDE [22], and SaWDE[13] algorithms on the datasets of Table 4. In these experiments, the results of the BDE-BSS algorithm show the effectiveness of the BSS algorithm in increasing the efficiency of the BDE algorithm. Moreover, the results of the BDE-BSS-DR algorithm show the effect of the DR mechanism in increasing the efficiency of the BDE-BSS algorithm. In addition, the results of the BDE-BSS-DR algorithm with the initialization based on the ReliefF algorithm have been examined, to show the impact of the search agents' initialization in increasing the efficiency. Therefore, BDE-BSS-DR represents the proposed algorithm with random initialization of search agents and BDE-BSS-DR-r represents the proposed algorithm with initialization based on ReliefF. In the initialization based on ReliefF, a step value is used to initialize the search agents. In the dataset with high dimensions, the step value is set to 20. Therefore, the first search agent is initialized with the top 20 features determined by ReliefF, the second search agent with the top 40 features, the third search agent with the top 60 features, and so on. In the low-dimensional and medium-dimensional datasets, the step value is set to 5. In the low-dimensional dataset, a number of search agents are initialized until reaching the number of features of the dataset. For example, in the Dermatology dataset, only 6 search agents and in the Z-Alizadeh Sani data set, only 10 search agents with the best feature determined by ReliefF are initialized, and the rest of the search agents are initialized randomly. The parameters of the proposed algorithms and other compared algorithms are given in Table 5.

In these experiments, the number of repetitions of the algorithms is set to 100 in low-dimensional and medium-dimensional datasets. Also, 200 is set in high-dimensional data. In Breast Cancer and COVID-19 datasets, which have more than 20,000 features, the number of iterations is set to 300. In addition, the number of iterations of the BSS algorithm in each call is set to 50. Therefore, for fair competition between algorithms in experiments with 100, 200, and 300 iterations, the number of function evaluations of all algorithms is 3000, 6000, and 9000, respectively. For example, in the proposed algorithms, assuming that the BSS algorithm is called once every five iterations in the worst case, it is called 18 times in 100 iterations. In each call, the evaluation function is called 50 times. Therefore, the evaluation function is called 900 times by the BSS algorithm. In addition, because there are 21 search agents, it is called 21 times in each iteration and 2100 times in 100 iterations. Therefore, the evaluation function is called 3000 times in one run. For this reason, in the BDE and the DimReM-BDE algorithms, the number of search agents is set to 30 so that the number of function evaluations in 100 iterations becomes 3000. On the other hand, in the SaWDE algorithm, the population size is 100. Therefore, every 30 function evaluations of this algorithm are considered one iteration.

In the fitness evaluation function, the KNN algorithm is used as a classifier due to its simplicity and effectiveness, where the Euclidean distance with K= 1 is used. Also, a five-fold cross-validation method has been used for training and testing. Therefore, the dataset is divided into five equal folds, and the classifier model is trained using four folds and evaluated using one remaining fold. This process is repeated five times, and finally, the average accuracy of the classification performed on the test sets is used as an evaluation criterion for obtaining the optimal subset of the features found by the search agents.

The number of independent runs to compare the efficiency of algorithms is 30. The average classification accuracy (Avg CA), the standard deviation of the classification accuracy (Std CA), the average running time, and the average number of selected features (Avg SF) are used as criteria for comparing the performance of algorithms.

The outcomes of 30 independent runs of the BDE-BSS-DR-r algorithm along with other algorithms on 20 datasets are presented in Table 6. The Wilcoxon rank-sum test with a significance level of 0.05 is used to assess the performance of the BDE-BSS-DR-r algorithm. The signifiers "+ ", "-", and "≈ " are used to indicate whether the performance of the BDE-BSS-DR-r algorithm is significantly better than, worse than, or equal to other compared algorithms. In addition, the Avg ranking of the Friedman test is also employed to identify significant improvements in the effectiveness of the BDE-BSS-DR-r algorithm when compared to other algorithms.

Comparing the results of BDE and BDE-BSS algorithms from Table 6 clearly shows the effect of the BSS algorithm in increasing the efficiency of the BDE algorithm. This superiority is especially evident in data with medium and high dimensions. The average results of the two algorithms show that the BSS algorithm has increased the accuracy of data classification by 11.35% compared to BDE. This shows that BSS has been able to achieve the desired goals, i.e., escaping from local optimal points and increasing efficiency. Comparing the results of two BDE-BSS and BDE-BSS-DR algorithms shows the importance of using the DR mechanism in increasing the efficiency of the BDE-BSS algorithm. The average results of the two algorithms show that the DR mechanism has increased the accuracy of data classification by 1.2% compared to BDE-BSS. The comparison of the two algorithms in data with high dimensions shows the importance of the DR mechanism. Therefore, this mechanism can be an effective method in increasing the efficiency of FS of high-

Table 6
The Avg CA and Std CA achieved through the applied methods.

No.	Dataset	BDE[46]	DimReM-BDE[22]	SaWDE[13]	BDE-BSS	BDE-BSS-DR	BDE-BSS-DR-r
1	Dermatology	98.66 ± 0.21 (≈)	98.52 ± 0.52 (≈)	98.83 ± 0.16 (≈)	98.74 ± 0.28 (≈)	98.64 ± 0.31 (≈)	98.68 ± 0.28
2	Spectf heart	83.59 ± 0.85 (+)	85.56 ± 0.82 (−)	84.94 ± 1.54 (≈)	84.83 ± 1.15 (≈)	84.88 ± 1.26 (≈)	85.08 ± 1.05
3	Z-Alizadeh Sani	87.63 ± 0.63 (+)	89.43 ± 0.99 (≈)	88.65 ± 0.94 (+)	88.83 ± 0.55 (+)	88.85 ± 0.67 (+)	89.29 ± 0.82
4	Scadi	87.57 ± 0.76 (+)	89.38 ± 1.03 (+)	90.50 ± 1.42 (≈)	89.85 ± 1.14 (+)	90.85 ± 1.43 (−)	90.28 ± 1.08
5	Arrhythmia	62.09 ± 0.84 (+)	64.94 ± 1.48 (+)	67.31 ± 2.23 (+)	66.72 ± 1.18 (+)	66.99 ± 1.22 (+)	68.82 ± 1.15
6	LSVT	93.65 ± 0.92 (+)	95.81 ± 0.72 (≈)	95.49 ± 1.03 (≈)	95.15 ± 1.19 (≈)	95.68 ± 0.93 (≈)	95.57 ± 1.09
7	Pomeroy-2002-v1	94.30 ± 2.11 (+)	97.11 ± 0.42 (+)	99.71 ± 0.87 (≈)	99.61 ± 1.24 (≈)	100.00 ± 0.0 (≈)	100.00 ± 0.0
8	Alizadeh-2000-v1	86.53 ± 1.18 (+)	87.50 ± 1.19 (+)	96.87 ± 2.83 (+)	98.42 ± 2.12 (+)	99.75 ± 0.76 (≈)	100.00 ± 0.0
9	Colon	79.30 ± 0.91 (+)	79.90 ± 0.99 (+)	95.74 ± 3.30 (+)	95.76 ± 1.50 (+)	96.85 ± 1.35 (+)	98.46 ± 1.06
10	SRBCT	89.76 ± 0.03 (+)	89.91 ± 0.63 (+)	99.12 ± 0.76 (+)	99.37 ± 0.86 (+)	100.00 ± 0.0 (≈)	100.00 ± 0.0
11	Leukemia	94.06 ± 0.82 (+)	94.25 ± 0.83 (+)	98.49 ± 0.53 (+)	98.56 ± 0.87 (+)	99.77 ± 0.50 (≈)	100.00 ± 0.0
12	Bladder Cancer	78.40 ± 2.48 (+)	78.40 ± 2.02 (+)	99.25 ± 0.39 (+)	98.83 ± 1.42 (+)	99.91 ± 0.45 (≈)	100.00 ± 0.0
13	DLBCL	89.76 ± 0.03 (+)	89.91 ± 0.63 (+)	97.79 ± 1.45 (+)	98.35 ± 1.85 (+)	99.91 ± 0.31 (≈)	100.00 ± 0.0
14	CNS	73.00 ± 1.10 (+)	73.27 ± 0.92 (+)	85.00 ± 3.16 (+)	89.61 ± 5.66 (+)	91.38 ± 3.06 (+)	98.38 ± 1.27
15	ALL-AML	88.90 ± 0.50 (+)	88.86 ± 0.51 (+)	98.72 ± 1.65 (+)	98.79 ± 1.58 (+)	99.71 ± 0.57 (≈)	100.00 ± 0.0
16	Brain_Tumor2	72.46 ± 1.45 (+)	72.73 ± 1.11 (+)	88.75 ± 3.41 (+)	95.40 ± 2.97 (+)	97.13 ± 2.14 (+)	99.53 ± 0.86
17	Prostate Tumor	89.02 ± 0.69 (+)	89.15 ± 0.86 (+)	94.01 ± 1.17 (+)	94.18 ± 1.73 (+)	97.09 ± 1.18 (+)	99.13 ± 0.49
18	MLL	90.41 ± 0.50 (+)	90.41 ± 0.61 (+)	96.12 ± 0.91 (+)	97.87 ± 1.70 (+)	100.00 ± 0.0 (≈)	100.00 ± 0.0
19	Breast Cancer	68.26 ± 0.52 (+)	67.49 ± 0.47 (+)	74.20 ± 1.25 (+)	88.93 ± 5.19 (+)	92.56 ± 1.89 (+)	97.33 ± 1.12
20	COVID-19	32.05 ± 0.45 (+)	32.66 ± 0.62 (+)	72.38 ± 2.02 (+)	88.62 ± 3.13 (+)	90.44 ± 3.27 (+)	94.00 ± 1.83
Wilcoxon Test (+ | ≈ | −)		19 | 1 | 0	16 | 3 | 1	15 | 5 | 0	16 | 4 | 0	8 | 11 | 1	-
Friedman Test (Avg Rank)		5.75	4.60	3.55	3.40	2.23	1.43
Avg		81.97	82.76	91.09	93.32	94.52	95.73

dimensional data.

Comparing the results of two algorithms BDE-BSS-DR and BDE-BSS-DR-r clearly shows the importance of initialization of search agents. The results show that the BDE-BSS-DR-r algorithm, in which the initialization of the search agents is performed using the superior features determined by the ReliefF algorithm, has an average of 1.21% data CA compared to BDE-BSS-DR. Due to the superiority of BDE-BSS-DR-r over the other two proposed algorithms, this algorithm is compared with other methods.

It is evident from Table 6 that the BDE-BSS-DR-r algorithm has a better average classification accuracy than the DimReM-BDE algorithm on 16 datasets out of a total of 20. Although both algorithms produce similar results in three datasets, DimReM-BDE outperforms BDE-BSS-DR on one dataset. Based on the outcomes of the statistical tests, the BDE-

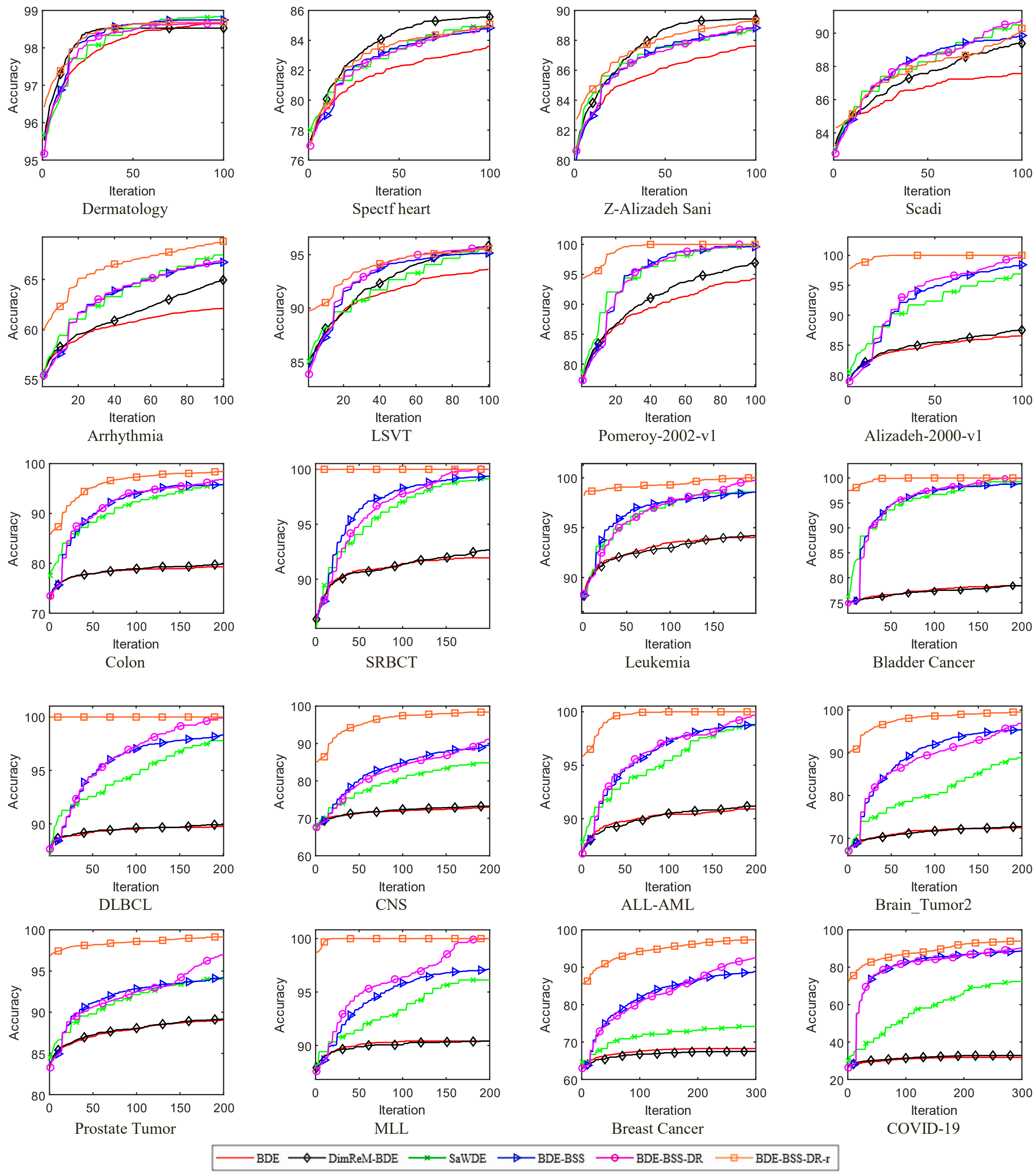


Fig. 3. The convergence diagram of the algorithms.

BSS-DR-r algorithm is superior to the DimReM-BDE algorithm. The comparison of the BDE-BSSE-DR-r algorithm to the DimReM-BDE algorithm indicates that the superiority of the BDE-BSS-DR-r is notably enhanced as the number of features increases. Based on the information presented in Table 6, it can be concluded that the BDE-BSS-DR-r algorithm outperforms the SaWDE algorithm in 15 of the 20 datasets. For the remaining five datasets, both methods show similar results. Results from both tests, Wilcoxon and Friedman, validate the superiority of the BDE-BSS-DR-r, which consistently performs better or is equal to the BDE algorithm. Although the BDE-BSS-DR-r algorithm is superior to SaWDE, in some datasets its superiority is minimal. The BDE-BSS-DR-r algorithm exhibits superior performance to the SaWDE algorithm by 0.88% in the SRBCT dataset, whereas in some datasets, the superiority is notably high. In particular, the BDE-BSS-DR-r algorithm demonstrates 10.78% and 21.62% superiority over the SaWDE algorithm in the Brain_Tumor2 and COVID-19 datasets, respectively. Comparing the efficiency of the BDE-BSS-DR-r algorithm with that of the BDE, DimReM-BDE and SaWDE indicates the superiority of 13.76%, 12.97% and 4.64% of these algorithms, respectively.

The convergence diagram depicted in Fig. 3 provides crucial information regarding the performance of the algorithms during the FS procedure. As we can see in Fig. 3, all algorithms have a similar search process in three low-dimensional datasets. The superiority of the three SaWDE, BDE-BSS, BDE-BSS-DR and BDE-BSS-DR-r algorithms over the two BDE and DimReM-BDE algorithms is evident in the five medium-dimension datasets. Also, in high-dimensional datasets, the remarkable superiority of the proposed algorithms over other BDE and DimReM-BDE algorithms is evident. Fig. 3 clearly illustrates that the BDE and DimReM-BDE algorithms exhibit reliable performance in high-dimensional spaces. The search process of the algorithms starts with a random point, then quickly converges to optimal solutions, and terminates at local optima. Moreover, as the number of features grows, these algorithms consistently outperform the SaWDE algorithm in the search process.

The behavior of the BDE-BSS and BDE-BSS-DR algorithms in most datasets, especially in medium-dimensional and high-dimensional datasets, shows that after 10 iterations when the BSS algorithm is called, this algorithm can dramatically increase the FS efficiency. As a result, the algorithm obtains better solutions from the other algorithms. Also, observing the behavior of the BDE-BSS-DR algorithm shows that after several calls of the DR mechanism in the final iterations of the algorithm, it has obtained a better solution than the BDE-BSS algorithm. This is because after calling the DR mechanism several times, the search space changes from a huge to a smaller search space, so the algorithm can get better solutions. For example, in the Breast Cancer dataset, with the DR mechanism, the BDE-BSS-DR algorithm was able to attain an average classification accuracy of 92.56%, while, the BDE-BSS achieved an average accuracy of 88.93% for classification. The BDE-BSS-DR algorithm also outperforms other algorithms with datasets with medium to high dimensions.

Observing the convergence process of the BDE-BSS-DR-r algorithm shows the significant impact of initialization at the beginning of the search process, especially in high-dimensional data. Therefore, it can be concluded that intelligent initialization of search agents, instead of random initialization, can play an important role in fast convergence and reaching the global optimal points of the algorithm. This issue is evident from the observation of the convergence process of the BDE-BSS-DR-r algorithm in most medium-dimensional data sets and all high-dimensional data sets. For example, the BDE-BSS-DR-r algorithm has reached the global optimal point in the MLL dataset with the least number of iterations, but the BDE-BSS-DR algorithm has achieved the global optimum in the last iterations.

Table 7 illustrates the average number of selected features by algorithms in 30 independent runs. Observing from the table, the BDE-BSS-DR-r algorithm has selected fewer features than the BDE algorithm in 19 out of 20 datasets. However, the outcomes of both algorithms are alike in the remaining dataset. Furthermore, the outcomes from the table indicate that on 17 out of 20 datasets, the BDE-BSS-DR-r algorithm selects a lesser number of features compared to the DimReM-BDE algorithm. On three datasets, the DimReM-BDE-r is better than the BDE-BSS-DR. With an increase in the number of features, it is evident that the BDE-BSS-DR-r is superior to the BDE and DimReM-BDE algorithms by a substantial margin, as substantiated by the results of the analysis. On average, the BDE and DimReM-BDE algorithms choose approximately 50% of the dataset's features. In contrast, the suggested techniques select multiple relevant features, regardless of the dataset's feature count. Based on the statistical test outcomes, it is evident that the BDE-BSS-DR-r algorithm outperforms both the BDE and DimReM-BDE algorithms, as concluded from the analysis. This is mainly evident in the case of medium and high-dimensional datasets, where the BDE-BSS-DR-r algorithm has tended to select fewer features than the other two algorithms.

According to Table 7, the BDE-BSS-DR-r algorithm outperforms the SaWDE algorithm on 15 out of 20 datasets. Analysts observed that in five

Table 7
Average number of features selected by each algorithm.

No.	Dataset	BDE [46]	DimReM-BDE [22]	SaWDE [13]	BDE-BSS	BDE-BSS-DR	BDE-BSS-DR-r
1	Dermatology	14.630 (−)	9.96000 (−)	10.000 (−)	15.100 (≈)	15.900 (≈)	15.930
2	SPECTF heart	18.200 (+)	11.8000 (−)	12.700 (−)	18.630 (+)	18.100 (+)	16.266
3	Z-Alizadeh Sani	25.330 (+)	16.3000 (−)	15.660 (−)	25.030 (+)	25.030 (+)	20.800
4	Scadi	98.000 (+)	64.4300 (+)	27.600 (−)	27.600 (−)	22.530 (−)	34.766
5	Arrhythmia	135.00 (+)	111.330 (+)	83.220 (+)	128.26 (+)	124.00 (+)	81.333
6	LSVT	151.60 (+)	130.000 (+)	85.260 (−)	152.40 (+)	103.10 (+)	86.933
7	Pomeroy-2002-v1	421.20 (+)	391.460 (+)	70.260 (+)	72.960 (+)	63.760 (+)	43.133
8	Alizadeh-2000-v1	543.80 (+)	509.760 (+)	104.06 (+)	39.400 (−)	48.600 (+)	21.866
9	Colon	988.10 (+)	936.500 (+)	59.430 (+)	25.380 (+)	22.500 (+)	20.666
10	SRBCT	1138.0 (+)	1083.00 (+)	134.50 (+)	74.560 (+)	91.200 (+)	25.120
11	Leukemia	2647.0 (+)	2587.00 (+)	212.00 (+)	187.33 (+)	110.50 (+)	24.060
12	Bladder Cancer	2820.0 (+)	2487.00 (+)	84.260 (+)	39.600 (−)	58.530 (+)	42.432
13	DLBCL	2695.0 (+)	2653.00 (+)	188.80 (+)	69.260 (+)	79.060 (+)	39.235
14	CNS	3531.0 (+)	3486.00 (+)	531.13 (+)	145.86 (+)	122.63 (+)	23.250
15	ALL-AML	3541.0 (+)	3493.00 (+)	133.40 (+)	94.230 (+)	90.960 (+)	17.231
16	Brain_Tumor2	5148.0 (+)	5109.00 (+)	155.20 (+)	30.300 (+)	50.630 (+)	18.866
17	Prostate Tumor	5214.0 (+)	5189.00 (+)	490.50 (+)	421.70 (+)	188.23 (+)	25.330
18	MLL	6205.0 (+)	6117.00 (+)	372.88 (+)	114.00 (+)	113.76 (+)	77.235
19	Breast Cancer	1217.0 (+)	12092.0 (+)	1103.0 (+)	124.23 (+)	113.30 (+)	51.533
20	COVID-19	12575 (+)	12203.0 (+)	498.50 (+)	12.960 (−)	28.230 (−)	38.412
Wilcoxon Test (+ | ≈ | −)		19 | 0 | 1	17 | 0 | 3	15 | 0 | 5	15 | 1 | 4	17 | 1 | 2	–
Friedman Test (Avg Rank)		5.70	4.35	3.28	3.15	2.73	1.80
Avg		2456.34	2934.03	218.62	90.94	74.53	36.22

Table 8
Average time consumed by algorithms in 30 independent runs per second.

No.	Dataset	BDE [46]	DimReM-BDE [22]	SaWDE [13]	BDE-BSS	BDE-BSS-DR	BDE-BSS-DR-r
1	Dermatology	111.50(−)	113.04(≈)	141.85(+)	109.45(−)	114.02(+)	112.78
2	SPECTF heart	109.61(−)	108.02(−)	120.11(≈)	107.61(−)	113.15(≈)	113.68
3	Z-Alizadeh Sani	110.71(−)	114.80(−)	121.09(+)	110.42(−)	117.42(+)	115.11
4	Scadi	107.69(−)	109.52(−)	115.12(+)	105.77(−)	111.88(≈)	112.10
5	Arrhythmia	153.48(+)	145.64(≈)	136.25(+)	141.64(−)	145.02(+)	148.76
6	LSVT	113.42(−)	109.44(−)	120.75(+)	111.66(−)	117.28(≈)	116.89
7	Pomeroy-2002-v1	105.22(−)	112.77(≈)	145.98(+)	102.90(−)	111.77(≈)	110.85
8	Alizadeh-2000-v1	113.02(+)	109.34(−)	150.28(+)	108.37(−)	111.40(≈)	110.25
9	Colon	283.90(+)	263.49(+)	573.56(+)	210.54(−)	213.05(≈)	212.82
10	SRBCT	259.09(+)	272.39(+)	557.90(+)	229.61(−)	231.18(≈)	230.63
11	Leukemia	354.06(+)	374.92(+)	2795.3(+)	260.18(≈)	261.02(+)	245.06
12	Bladder Cancer	290.91(+)	283.85(+)	2603.2(+)	229.09(+)	224.78(+)	219.34
13	DLBCL	389.24(+)	384.27(+)	2310.3(+)	261.32(+)	246.59(+)	238.15
14	CNS	380.46(+)	477.13(+)	4644.8(+)	270.74(+)	264.62(+)	251.58
15	ALL-AML	368.23(+)	358.60(+)	3879.9(+)	275.76(−)	281.85(+)	258.85
16	Brain_Tumor2	402.25(+)	405.02(+)	10716 (+)	266.54(−)	274.59(+)	254.31
17	Prostate Tumor	593.25(+)	602.26(+)	9207 (+)	384.63(+)	348.63(+)	339.52
18	MLL	554.35(+)	526.92(+)	12224 (+)	357.13(+)	311.97(+)	335.02
19	Breast Cancer	1410.1(+)	1382.0(+)	50265 (+)	733.56(+)	621.74(+)	550.74
20	COVID-19	1475.8(+)	1432.5(+)	48325 (+)	635.84(+)	538.24(+)	480.30
Wilcoxon Test (+ | ≈ | −)		14 | 0 | 6	12 | 3 | 5	19 | 1 | 0	12 | 1 | 7	13 | 7 | 12	
Friedman Test (Mean Rank)		4.00	3.84	5.75	1.95	3.10	2.35
Avg		384.31	384.29	7457.67	250.63	238.01	227.83

of these datasets, the SaWDE algorithm selects fewer features than the BDE-BSS-DR-r algorithm. Moreover, the results from both the Wilcoxon and Friedman tests also indicate that BDE-BSS-DR-r is superior to the SaWDE algorithm since it achieves significantly better performance in most datasets. The BDE-BSS-DR-r algorithm has been found superior when compared to the BDE-BSS in 15 out of 20 datasets. In one dataset, both algorithms have produced similar results, while the BDE-BSS algorithm outperformed the BDE-BSS-DR-r algorithm in four datasets. The comparison of two algorithms, BDE-BSS-DR and BDE-BSS-DR-r, shows the superiority of the BDE-BSS-DR-r algorithm in terms of the average number of selected features. This advantage is due to the initialization of the search agents.

Overall, the comparison of all three proposed algorithms with other methods in terms of average accuracy and the Friedman test shows that these methods have the first to third ranks. Therefore, all three presented methods, BSS, DR mechanism and initialization of search agents using the best features determined by the ReliefF algorithm, play a crucial role in reducing the number of selected features.

The runtime of FS algorithms is influenced by several factors, such as the running time of the techniques used and the time taken for fitness

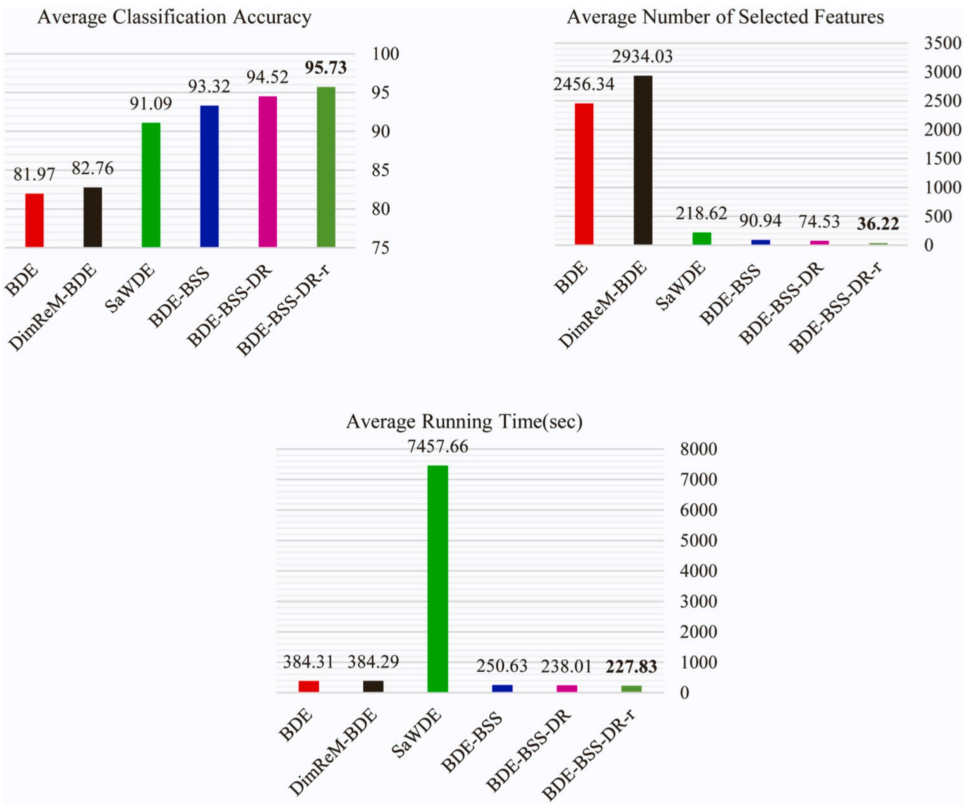


Fig. 4. Average classification accuracy, number of selected features and running time(sec) of the algorithms.

function evaluation. For instance, certain FS algorithms employ filter-based techniques to identify relevant features and minimize the search space dimensions on the cost of running time. Notably, the runtime of the fitness function evaluation is directly proportional to the number of features in the dataset. Thus, evaluating solutions with a large number of features consumes more time than solutions with fewer features. Therefore, an FS algorithm that achieves high classification accuracy with fewer features leads to a significant reduction in computational costs. In Table 8, the average runtime of algorithms in 30 independent runs per second for each dataset is displayed. The data suggest that the proposed algorithms perform at a higher speed compared to other algorithms. The BDE-BSS-DR-r algorithm has demonstrated superiority over other algorithms as confirmed by the Wilcoxon test. However, it should be noted that the DR mechanism increases the runtime of the algorithm. Nonetheless, removing unimportant features from both datasets and search agents via the DR mechanism results in a substantial decrease in the computational cost of the fitness evaluation function, particularly in high-dimensional datasets. This issue becomes more pronounced in datasets with a high number of dimensions. Furthermore, the use of the BSS algorithm for FS, coupled with the exclusion of a large number of insignificant features during the first stages of the search process, can considerably reduce the cost of computations, particularly in medium and high-dimensional data.

Overall, the initialization of the search agents using the most important features determined by the ReliefF algorithm causes the search agents to have much fewer features than the random initialization method, and as a result, the time of evaluating the search agents will decrease. This issue is evident by comparing the running time of BDE-BSS-DR and BDE-BSS-DR-r algorithms, especially in high-dimensional data. Fig. 4 displays the average classification accuracy, number of selected features and running time(sec) by six algorithms across 20 datasets. The comparison of the results of the three proposed algorithms and other algorithms shows that the BDE-BSS-DR-r algorithm has the highest average classification accuracy, the lowest number of selected features and the shortest execution time algorithm among the algorithms.

4.3.2. Comparative analysis of the proposed algorithm with other algorithms

In this section, the efficiency of the proposed BDE-BSS algorithm as a wrapper-based FS algorithm is compared with that of the SFE algorithm [3], which is one of the latest wrapper-based FS algorithms presented for feature selection of high-dimensional data. In addition, the efficiency of the BDE-BSS-DR-r algorithm as a hybrid FS algorithm is compared with efficiency of a hybrid algorithm called VGS-MOEA [30]. The experiments performed in this section are different from the experiments performed in previous section. In this section, each dataset is randomly divided into training (80%) and test (20%) sets. All the algorithms perform the FS process using the same training set. After the FS process, the performance of the algorithms are compared on the test dataset (unseen data). In the evaluation function of all algorithms, KNN classifier with K= 5 and Euclidean distance are used. In addition, the five-fold cross-validation method is used to divide the training set (80%) into two training and test sets to evaluate the solutions obtained by the algorithms. In these experiments, the parameters suggested by the author of SFE [3] and VGS-MOEA [29] algorithms have been used. In addition, in the FS process of each algorithm, the evaluation function is called 5000 times in order to have the same evaluation condition. Each of the algorithms is run 30 times independently on the same training and test sets and their results are compared. Table 9 presents the results of the algorithms on the training and testing datasets.

Comparing the results of the two wrapper-based SFE algorithms and the proposed wrapper-based BDE-BSS algorithm based on the Wilcoxon Test and the average of different classification criteria presented in Table 9 shows that the proposed BDE-BSS algorithm has a better performance than the SFE algorithm. Nevertheless, the SFE algorithm has achieved significant superiority over the BDE-BSS algorithm regarding the average running time and the number of selected features. On the other hand, as we can see in Table 9, the BDE-BSS-DR-r algorithm shows a very impressive performance in the feature selection of the training and test sets compared to the SFE and BDE-BSS algorithms. This superiority is evident by comparing the results of the algorithms in the Wilcoxon Test and the average results of the algorithms. This issue firstly shows the effect of using BSS and DR methods in selecting features from high-dimensional data, and secondly, it shows the effect of filter-based algorithms in increasing the efficiency of hybrid algorithms.

The comparison of two hybrid algorithms BDE-BSS-DR-r and VGS-MOEA in the training dataset shows that the BDE-BSS-DR-r algorithm has a significant advantage over the VGS-MOEA algorithm. This superiority is evident by observing the results of the Wilcoxon test and the average results of the algorithms in different evaluation criteria. Moreover, regarding the results obtained from test set, the BDE-BSS-DR-r algorithm has an acceptable advantage over the VGS-MOEA algorithm. The running time of both algorithms is almost the same, however, the remarkable point about the running time of the algorithms is that with the increase in the dimension of the data (for example, the datasets of COVID-19 and Breast Cancer), the running time of the BDE-BSS-DR-r algorithm is reduced, compared to running time of VGS-MOEA algorithm. The reason is that the DR mechanism reduces the search time of the algorithm by reducing the dimensions of the search space. Also, the results of Table 9 show that the BDE-BSS-DR-r algorithm has significantly better performance than the VGS-MOEA algorithm in terms of the number of selected features.

In general, the results of this section and the previous section show that the proposed algorithms have high efficiency compared to other FS algorithms, which is due to the use of BSS algorithms and DR mechanism. Therefore, these methods can be used for classification and gene selection applications.

4.3.3. Application of BDE-BSS in selecting the feature of heart disease

Coronary artery disease, a type of cardiovascular disease, is responsible for causing the death of over 17.9 million people each year, which accounts for approximately 31% of all global deaths. Several factors such as age, gender, high blood pressure, obesity, being overweight, elevated blood cholesterol levels, and maintaining a healthy lifestyle are critical to mitigate the risk of this disease [47,48,49]. Determining the principal risk factors and using diagnostic systems can play a decisive role in reducing mortality caused by cardiovascular diseases. Appropriate systems can be designed to diagnose these diseases on time using FS algorithms. It should be noted that the BDE-BSS-DR algorithm is for FS in datasets with high dimensions, but its efficiency in low-dimensional datasets is similar to the BDE-BSS algorithm. Therefore, this section evaluates only the BDE-BSS algorithm on the heart disease datasets collected by Alizadeh Sani [49]. There is information on 303 people in this dataset, of which 216 people are sick with coronary artery disease and 87 people are healthy [48,49].

The SVM with a polynomial kernel and a scale of 5.12 is selected after a comprehensive parameter tuning process. The evaluation approach used is the 10-fold cross-validation. The size of the population is set to 100 for the BDE-BSS algorithm. In addition, the number of iterations of the BDE-BSS and BSS algorithms is set to 200 and 30, respectively. The proposed algorithm is executed 30 times to ensure the reliability of the results. The outcomes are provided in Table 10. The average, the best performance, and the worst performance with different evaluation criteria are shown. The proposed algorithm demonstrated its high accuracy with a top-performing classification rate of 95.05%, and even in the worst-case scenario, the accuracy remained at 94.72%. This comparison highlights the algorithm's stability and its ability to consistently identify optimal subsets.

The confusion matrices, ROC curve, and convergence curves of the best run can be found in Appendix 4. In addition, the features' name, their values and 20 features selected by the BDE-BSS algorithm can be

Table 9
The results of proposed algorithms and compared algorithms.

Dataset		SFE [3]		BDE-BSS		VGS-MOEA [30]		BDE-BSS- DR-r	
	Metric	**Train (80%)**	**Test (20%)**	**Train (80%)**	**Test (20%)**	**Train (80%)**	**Test (20%)**	**Train (80%)**	**Test (20%)**
Colon	Accuracy	92.26±2.81 (+)	**82.22±8.85 (−)**	92.93±3.29 (+)	81.11±7.25 (+)	93.06±1.82 (+)	81.11±7.19 (+)	**94.53 ±1.73**	81.66±5.33
	Recall	93.53±7.20 (+)	77.44±17.2 (+)	92.86±5.58 (+)	81.47±15.5 (+)	91.88±5.76 (+)	78.00±16.2 (+)	**94.22 ±5.11**	**84.47 ±15.6**
	Precision	79.89±6.10 (+)	66.66±18.9 (−)	82.73±6.89 (≈)	63.33±18.8 (≈)	**90.34±6.31 (−)**	**68.33±17.9 (−)**	82.45±6.03	63.33±16.8
	F1_score	86.57±6.26 (+)	**69.73±16.2 (−)**	87.04±5.21 (+)	66.79±15.2 (+)	88.10±6.12 (+)	69.40±12.1 (−)	**90.07 ±5.11**	68.56±10.3
	Time (sec)	**178 (−)**	–	207 (+)	–	188 (−)	–	204	–
	Avg SF	**13.13 (−)**	–	18.8 (−)	–	23.06 (+)	–	19.33	–
	Accuracy	98.52±1.46 (+)	81.66±8.11 (+)	98.91±2.13 (+)	82.50±2.82 (+)	**100.0±0.00 (≈)**	91.25±5.11 (+)	**100±0.0**	**95.41 ±1.11**
	Recall	98.81±1.25 (+)	84.89±7.21 (+)	99.08±1.77 (+)	86.14±4.11 (+)	**100.0±0.00 (≈)**	93.08±3.44 (+)	**100±0.0**	**96.92 ±1.59**
SRBCT	Precision	**100.0±0.00 (≈)**	85.05±6.32 (+)	98.70±2.32 (+)	84.94±3.69 (+)	**100.0±0.00 (≈)**	91.83±5.21 (+)	**100±0.0**	**96.16 ±1.58**
	F1_score	98.59±1.47 (+)	82.62±8.45 (+)	98.94±2.08 (+)	83.67±3.84 (+)	**100.0±0.00 (≈)**	91.09±5.58 (+)	**100±0.0**	**96.20 ±1.36**
	Time (sec)	**178 (−)**	–	222 (+)	–	192 (−)	–	209	–
	Avg SF	**31.86 (−)**	–	274.6 (+)	–	74.33 (−)	–	85.33	–
	Accuracy	95.39±3.12 (+)	80.95±6.98 (+)	96.65±2.31 (+)	83.80±7.28 (+)	**99.64±0.73 (≈)**	**91.42±5.62 (−)**	99.53±0.79	90.00±7.86
	Recall	95.30±3.99 (+)	78.83±11.7 (+)	97.16±3.12 (+)	85.91±12.1 (+)	**99.71±2.14 (−)**	**88.81±12.1 (≈)**	99.20±2.14	88.78±11.7
Leukemia	Precision	95.71±4.28 (−)	75.96±13.1 (+)	95.89±4.71 (−)	82.05±12.2 (+)	**99.77±0.86 (−)**	**89.75±9.12 (≈)**	93.98±7.20	89.47±10.1
	F1_score	93.86±4.02 (+)	73.79±14.3 (+)	96.10±3.78 (+)	82.43±12.0 (+)	**99.58±0.91 (−)**	**88.38±11.2 (≈)**	98.97±1.54	88.10±11.0
	Time (sec)	**183 (−)**	–	250 (+)	–	205 (−)	–	220	–
	Avg SF	45.46 (−)	–	320.2 (+)	–	76.06 (+)	–	**42.25**	–
Bladder Cancer	Accuracy	92.22±4.75 (+)	69.16±12.2 (+)	98.15±2.52 (+)	66.66±14.1 (+)	98.82±1.84 (+)	74.16±11.9 (+)	**99.58 ±1.09**	**80.00 ±9.11**
	Recall	90.87±6.13 (+)	61.22±20.1 (+)	98.07±3.12 (+)	60.11±18.7 (+)	99.17±2.58 (≈)	71.96±16.3 (+)	**99.22 ±2.58**	**78.62 ±14.8**
	Precision	94.29±5.62 (+)	63.51±18.4 (+)	99.82±0.68 (−)	63.88±15.8 (+)	**100.0±0.00 (−)**	71.85±14.7 (+)	97.19±2.52	**78.14 ±11.9**
	F1_score	90.24±5.35 (+)	58.79±17.7 (+)	97.65±3.53 (+)	59.71±16.0 (+)	98.75±1.88 (+)	69.37±15.9 (+)	**99.25 ±2.32**	**75.16 ±13.2**
	Time (sec)	**179 (−)**	–	224 (+)	–	196 (−)	–	212	–
	Avg SF	**31 (−)**	–	40.93 (+)	–	61 (+)	–	51.1	–
	Accuracy	97.48±1.91 (+)	85.06±7.02 (+)	98.37±1.05 (+)	83.73±7.81 (+)	99.93±0.35 (≈)	91.73±7.01 (+)	**99.81 ±0.47**	**89.86 ±6.26**
	Recall	97.24±2.15 (+)	90.15±6.79 (+)	98.51±1.07 (+)	89.68±6.91 (+)	99.92±0.38 (≈)	95.37±4.21 (+)	**99.85 ±0.38**	**92.34 ±4.93**
DLBCL	Precision	**100.0±0.00 (−)**	90.90±6.95 (+)	98.94±2.34 (+)	89.72±9.74 (+)	**100.0±0.00 (−)**	93.96±6.35 (+)	99.28±2.70	**94.72 ±5.91**
	F1_score	98.41±1.21 (+)	90.14±4.51 (+)	98.94±0.69 (+)	89.08±5.97 (+)	99.95±0.22 (≈)	94.43±5.71 (+)	**99.87 ±0.31**	**93.29 ±3.88**
	Time (sec)	**195 (−)**	–	248 (+)	–	229 (+)	–	219	–
	Avg SF	**34.44 (−)**	–	87.4 (+)	–	78.76 (+)	–	25.12	–
	Accuracy	85.80±2.25 (+)	53.75±9.89 (+)	87.15±2.98 (+)	62.50±8.21 (+)	93.17±1.95 (+)	**64.58±7.52 (≈)**	**96.57 ±0.56**	**64.58 ±7.86**
	Recall	86.84±3.01 (+)	62.38±11.1 (+)	89.43±2.29 (+)	69.11±8.11 (−)	94.66±1.75 (+)	**69.91±8.12 (−)**	**95.34 ±1.23**	68.80±8.35
CNS	Precision	87.89±1.86 (+)	66.33±9.51 (+)	88.65±1.96 (+)	75.08±8.01 (+)	93.61±2.56 (+)	78.30±9.69 (+)	**100.0 ±0.00**	**82.94 ±6.82**
	F1_score	86.49±2.11 (+)	63.57±9.69 (+)	89.76±1.85 (+)	71.02±8.18 (+)	94.51±2.26 (+)	73.49±8.36 (+)	**97.44 ±0.84**	**74.63 ±7.58**
	Time (sec)	**198** (−)	–	276 (+)	–	221 (−)	–	238	–
	Avg SF	33.65 (+)	–	240.3 (+)	–	88.8 (+)	–	**31.45**	–
	Accuracy	97.53±2.18 (+)	83.21±7.85 (+)	98.61±1.84 (+)	85.35±0.82 (+)	99.81±0.55 (≈)	**91.07±6.22 (−)**	**99.91 ±0.37**	88.57±7.12
	Recall	99.30±1.71 (+)	84.75±15.2 (+)	99.75±1.11 (+)	90.14±0.82 (−)	**100.0±0.00 (≈)**	**94.66±8.18 (−)**	**100.0 ±0.00**	89.25±10.3
ALL-AML	Precision	94.62±9.51 (+)	65.00±16.1 (+)	94.07±9.65 (+)	70.00±0.82 (+)	98.38±7.22 (+)	**80.00±14.1 (−)**	**100.0 ±0.00**	78.00±15.2

(continued on next page)

Table 9 (*continued*)

Dataset		SFE [3]		BDE-BSS		VGS-MOEA [30]		BDE-BSS- DR-r	
	Metric	**Train (80%)**	**Test (20%)**	**Train (80%)**	**Test (20%)**	**Train (80%)**	**Test (20%)**	**Train (80%)**	**Test (20%)**
	F1_score	96.15±2.30 (+)	72.60±14.5 (+)	97.80±6.53 (+)	75.24±0.82 (+)	99.60±1.23 (≈)	**85.68±10.3 (−)**	**99.88 ±0.49**	82.39±11.6
	Time (sec)	**171** (−)	–	241 (+)	–	201 (−)	–	213	–
	Avg SF	30.45 (+)	–	62.6 (+)	–	50 (+)	–	**13.1**	–
	Accuracy	85.00±4.90 (+)	**71.33±7.91 (−)**	88.50±3.75 (+)	65.33±14.6 (+)	88.00±3.56 (+)	66.66±13.3 (+)	**96.33 ±2.25**	68.66±12.2
	Recall	84.77±6.32 (+)	**68.31±13.1 (−)**	88.80±5.96 (+)	61.88±18.1 (≈)	91.85±2.95 (+)	64.36±16.6 (−)	**96.83 ±2.95**	61.86±20.2
Brain_ Tumor2	Precision	95.59±3.30 (+)	**69.30±6.71 (−)**	98.23±1.44 (+)	63.61±14.2 (+)	98.31±1.17 (+)	64.44±13.4 (+)	**98.77 ±1.66**	65.55±17.1
	F1_score	82.42±6.06 (+)	**65.52±9.62 (−)**	86.99±5.35 (+)	59.68±16.6 (+)	89.20±3.56 (+)	61.67±14.8 (−)	**96.38 ±3.04**	60.69±17.3
	Time (sec)	**179** (−)	–	253 (+)	–	208 (−)	–	226	–
	Avg SF	**35.73** (+)	–	68.33 (+)	–	134.2 (+)	–	33.33	–
Prostate Tumor	Accuracy	94.38±2.30 (+)	84.00±6.51 (+)	95.10±2.12 (+)	86.75±4.91 (+)	96.87±1.13 (+)	90.50±4.82 (+)	**98.68 ±0.86**	**91.75 ±4.21**
	Recall	94.86±3.28 (+)	85.15±8.01 (+)	96.38±2.84 (+)	87.91±7.85 (+)	98.30±1.01 (+)	92.68±6.95 (≈)	**99.27 ±1.01**	**92.82 ±5.39**
	Precision	85.86±7.51 (+)	83.51±10.6 (+)	90.52±4.71 (+)	86.56±6.15 (+)	86.36±7.40 (+)	89.00±7.89 (+)	**100.0 ±0.00**	**91.50 ±6.98**
	F1_score	94.61±2.09 (+)	83.69±7.56 (+)	95.19±2.01 (+)	86.75±4.98 (+)	96.88±1.18 (+)	90.31±4.74 (+)	**98.69 ±0.83**	**91.60 ±4.17**
	Time (sec)	**176** (−)	–	317 (+)	–	231 (+)	–	221	–
	Avg SF	45.75 (+)	–	213 (+)	–	134.5 (+)	–	**44.15**	–
	Accuracy	96.47±2.88 (+)	82.85±8.06 (+)	96.89±2.71 (+)	87.61±6.14 (+)	99.43±0.82 (+)	90.95±4.86 (+)	**100±0.0**	**95.23 ±3.83**
	Recall	96.72±2.90 (+)	85.42±6.26 (+)	97.20±2.17 (+)	88.73±6.66 (+)	99.41±0.40 (+)	91.87±4.17 (+)	**100±0.0**	**96.66 ±2.91**
MLL	Precision	99.75±0.34 (+)	82.59±7.42 (+)	99.68±0.62 (+)	87.18±7.57 (+)	**100.0±0.00** (≈)	**95.75±4.88 (−)**	**100±0.0**	94.77±4.11
	F1_score	95.95±3.37 (+)	81.53±8.92 (+)	96.72±2.78 (+)	86.97±7.11 (+)	99.37±0.92 (+)	90.62±4.91 (+)	**100±0.0**	**94.98 ±4.03**
	Time (sec)	**183** (−)	–	322 (+)	–	320 (+)	–	250	–
	Avg SF	**54.66** (−)	–	231 (+)	–	56.8 (−)	–	101.3	–
	Accuracy	84.21±2.53 (+)	54.38±7.25 (+)	84.16±3.48 (+)	57.89±7.53 (+)	90.07±2.26 (+)	**64.21±7.32 (−)**	**95.00 ±1.95**	63.15±9.76
	Recall	81.93±3.29 (+)	56.38±7.32 (+)	83.42±4.46 (+)	60.28±7.81 (+)	93.00±2.50 (+)	**67.06±7.85 (−)**	**95.10 ±2.50**	63.68±9.36
Breast Cancer	Precision	97.78±1.94 (+)	68.00±12.8 (+)	98.49±0.94 (+)	60.66±11.3 (+)	98.52±2.28 (+)	64.66±12.9 (+)	**99.76 ±0.50**	**72.66 ±13.1**
	F1_score	85.89±2.79 (+)	60.64±6.89 (+)	85.35±3.16 (+)	59.95±8.24 (+)	90.11±2.14 (+)	64.95±8.58 (+)	**95.22 ±1.89**	**67.22 ±9.01**
	Time (sec)	**208** (−)	–	589 (+)	–	339 (+)	–	325	–
	Avg SF	108 (+)	–	627.0 (+)	–	184.4 (+)	–	**70.93**	–
	Accuracy	74.92±5.35 (+)	51.85±8.24 (+)	75.07±9.25 (+)	55.55±11.3 (+)	77.34±4.47 (+)	**64.44±9.20 (−)**	**87.80 ±2.24**	64.07±7.25
	Recall	77.30±5.45 (+)	52.41±8.55 (+)	76.06±7.71 (+)	59.75±9.13 (+)	78.39±2.89 (+)	**64.97±12.6 (−)**	**88.63 ±2.81**	61.41±10.8
COVID-19	Precision	83.97±2.00 (+)	51.85±5.12 (+)	97.83±0.95 (+)	55.55±11.3 (+)	96.96±1.39 (+)	**64.44±9.20 (−)**	**98.32 ±0.71**	64.07±7.22
	F1_score	73.44±6.52 (+)	51.18±7.14 (+)	73.22±8.98 (+)	55.82±11.2 (+)	75.02±5.58 (+)	**61.97±9.90 (≈)**	**86.20 ±3.83**	61.94±8.92
	Time (sec)	**219** (−)	–	535 (+)	–	346 (+)	–	331	–
	Avg SF	100.5 (+)	–	498.7 (+)	–	374.4 (+)	–	**96**	–
	Accuracy	12 | 0 | 0	10 | 0 | 2	12 | 0 | 0	12 | 0 | 0	8 | 0 | 4	7 | 1 | 4	–	–
	Recall	12 | 0 | 0	11 | 0 | 1	12 | 0 | 0	12 | 0 | 6	7 | 1 | 4	5 | 2 | 5	–	–
(+ | ≈ | −)	Precision	9 | 1 | 2	10 | 0 | 2	9 | 1 | 2	11 | 1 | 0	6 | 2 | 4	7 | 4 | 1	–	–
	F1_score	11 | 0 | 1	10 | 0 | 2	12 | 0 | 0	9 | 1 | 2	8 | 3 | 1	7 | 3 | 2	–	–
	Time (sec)	0 | 0 | 12	–	12 | 0 | 0	–	5 | 0 | 7	–	–	–
	Avg SF	6 | 0 | 6	–	12 | 0 | 0	–	10 | 0 | 2	–	–	–
	Accuracy	91.18	73.37	92.54	74.90	94.68	80.17	**97.31**	**81.08**
	Recall	91.46	73.94	93.06	76.76	95.52	81.06	**97.31**	**81.30**
Avg	Precision	92.95	72.39	95.30	73.55	96.85	79.36	**97.48**	**80.94**
	F1_score	90.22	71.15	91.98	73.09	94.26	78.45	**96.83**	**79.56**
	Time (sec)	**187.25**	–	307.00	–	239.67	–	239.00	–
	Avg SF	**46.79**	–	223.57	–	111.36	–	51.67	–

Table 10
Comparison between BDE-BSS and BDE algorithm.

Algorithm		Accuracy	Precision	Recall	F1-Score	ROC
BDE-BSS	Best	95.05	96.33	96.82	96.58	0.9426
	Mean	94.91	96.33	96.53	96.43	0.9412
	Worst	94.72	96.33	96.32	96.32	0.9401
BDE	Best	94.72	96.33	96.32	96.32	0.9401
	Mean	94.34	95.01	95.11	95.06	0.9321
	Worst	93.73	94.31	94.43	94.37	0.9321

found in Appendix 5. Column SF represents the features and factors influencing coronary heart disease, determined by the BDE-BSS FS algorithm. A comparison between the suggested BDE-BSS and many state-of-the-art on the same dataset is presented in Table 11. The proposed method for selecting the most significant features in coronary artery disease has performed better than the methods presented in recent years. Table 11 shows that the proposed method using the 20 features has achieved 95.05% classification accuracy and 96.58% F1-score, significantly better than other methods presented in recent years. The proposed approach's superiority over other methods is the use of BSS in the BDE algorithm search process. In addition, SVM with different kernels is examined. Finally, the SVM classifier with a polynomial kernel and a scale of 5.12 is chosen for the FS, which effectively classified the Z-Alizadeh Sani coronary artery disease (CAD) dataset.

4.3.4. *Application of BDE-BSS and BDE-BSS-D-r in gene selection*

FS algorithms are successfully employed in gene selection from gene expression microarray datasets. The process of selecting pertinent genes from a large pool of over a thousand genes can significantly contribute to identifying the underlying causes of various diseases, including cancer. Additionally, this gene selection process can aid in the development of disease diagnostic systems with improved accuracy and efficacy. However, Gene expression data generated by DNA microarray technology is a high-dimensional, low-sample dataset containing many unimportant features. Therefore, FS algorithms in gene selection of this dataset face problems such as stopping the algorithms at local optimal and the curse of dimensionality. Efficient FS algorithms are necessary to address these issues. As the BDE-BSS-DR and BDE-BSS-DR-r have outperformed the BDE-BSS approach in prior experiments, only the BDE-BSS-DR and BDE-BSS-DR-r outcomes are utilized in this section for gene selection from 11 cancer datasets and one COVID-19 disease dataset.

Furthermore, we evaluate the outcomes obtained from the BDE-BSS-DR and BDE-BSS-DR-r algorithms in contrast to the approaches introduced in recent years. The fitness evaluation function of the proposed algorithms uses SVM classifier with a linear kernel and a five-fold cross-validation method. Furthermore, the proposed algorithms run for a total of 500 iterations while the BSS runs for 50 iterations. The DR mechanism from 150 iterations onwards is called if the algorithm fails to improve performance in 50 consecutive iterations. The proposed algorithm is run on each dataset 15 times, and the results are compared with that of the MRMR-DBH [58], Robust Minimum Redundancy Maximum Relevancy-Modified Gray Wolf Optimizer (rMRMR-MGWO) [59], Mutual Information Maximization – modified Moth Flame Algorithm (MIM-mMFA) [60] and Quantum Moth Flame Optimization Algorithm (QMFOA) [61] algorithms. The results of MRMR-DBH, rMRMR-MGWO, MIM-Mmfa, and QMFOA algorithms are taken from the original articles. A comparison between the proposed BDE-BSS-DR, BDE-BSS-DR-r and many recent advanced algorithms is provided in Table 12.

As both BDE-BSS-DR and BDE-BSS-DR-r algorithms have almost the same results (with the superiority of the BDE-BSS-DR-r algorithm in just two data sets), the results of the BDE-BSS-DR algorithm have just been analyzed in this section. As indicated in Table 12, the proposed algorithms have attained 100% classification accuracy in its best run across all 12 datasets. Notably, even in datasets such as Breast Cancer and COVID-19, which possess a substantial number of genes (24,481 and 25,534 genes, respectively), the suggested algorithms have achieved a remarkably better accuracy of classification compared to the algorithms proposed by other researchers. Moreover, the proposed algorithms have achieved 100% average classification accuracy in all datasets except for Breast Cancer and COVID-19 datasets. This implies that the proposed

Table 11
Comparison of the SVM+BDE-BSS method with other methods on the Z-Alizadeh Sani dataset.

No.	Reference	Year	Method	No. of SF	Accuracy	F1-Score
1	[49]	2013	SMO - Information Gain+ SVM Weights (FS)	21	94.08	–
2	[50]	2017	ANN-GA-SVM Weights	22	93.85	–
3	[51]	2017	SVM+ EA-MFS	34	93.70	95.53
4	[52]	2017	SVM	27	86.67	–
5	[48]	2019	2Genetic-nuSVM	29	93.08	91.51
6	[53]	2019	NE-nu-SVC+ multi-step balancing	–	94.66	–
7	[54]	2020	Two-tier ensemble + PSO	27	91.18	90.91
8	[55]	2020	CART	5	92.41	–
9	[56]	2020	Emotional Neural Network+PSO	22	88.34	92.12
10	[57]	2021	SVM+XG-Boost+ RF	–	93.86	–
11	Proposed approach	–	SVM+BDE-BSS	20	**95.05**	**96.58**

Table 12
The results of proposed algorithms and compared algorithms in the application of gene selection.

Dataset	MRMR-DBH [57]		rMRMR-MGWO [59]		MIM-mMFA [60]		QMFOA [61]		BDE-BSS-DR		BDE-BSS-DR-r	
	Best	Avg (Std)	Best	Avg (Std)	Best	Avg (Std)	Best	Avg (Std)	Best	Avg (Std)	Best	Avg (Std)
Colon	98	97.02 ± 0.	–	95.8	100	100 ± 0.0	100	100 ± 0.0	100	100 ± 0.0	100	100 ± 0.0
SRBCT	–	–	–	100	100	99.40 ± 0.6	100	99.40 ± 0.9	100	100 ± 0.0	100	100 ± 0.0
DLBCL	100	100 ± 0.0	–	–	100	100 ± 0.0	–	–	100	100 ± 0.0	100	100 ± 0.0
Leukemia	–	–	–	–	–	–	100	100 ± 0.0	100	100 ± 0.0	100	100 ± 0.0
Bladder Cancer	–	–	–	–	–	–	–	–	100	100 ± 0.0	100	100 ± 0.0
CNS	97.19	95.4 ± 1.5	–	99.3	100	99.83 ± 0.5	100	100 ± 0.0	100	100 ± 0.0	100	100 ± 0.0
ALL-AML	100	100 ± 0.0	–	100	–	–	–	–	100	100 ± 0.0	100	100 ± 0.0
Brain_Tumor2	–	–	–	–	100	100 ± 0.0	100	100 ± 0.0	100	100 ± 0.0	100	100 ± 0.0
Prostate Tumor	99.6	98.19 ± 0.3	–	–	100	100 ± 0.0	100	99.87 ± 0.5	100	100 ± 0.0	100	100 ± 0.0
MLL	100	100 ± 0.0	–	100	100	100 ± 0.0	–	–	100	100 ± 0.0	100	100 ± 0.0
Breast Cancer	93.8	90.21 ± 2.2	–	–	91.7	86.80 ± 3.1	81.44	77.53 ± 2.0	100	99.18 ± 1.49	100	99.93 ± 0.2
COVID-19	83.33	–	–	–	–	–	–	–	100	97.72 ± 2.02	100	99.40 ± 0.8

Table 13
The results of the selected genes in one of the best runs and the average number of selected genes on 15 runs by the BDE-BSS-DR algorithm.

Dataset	Selected genes index in the best run of the BDE-BSS-DR algorithm	Average number of selected genes
Colon	104, 195, 353, 413, 513, 652, 799, 939, 986, 1246, 1348, 1482, 1493, 1644: **(14 Genes)**	26.9
SRBCT	445, 779, 971, 976, 1319, 1488, 1575, 1893, 1955, 2186, 2304: **(11 Genes)**	26.8
DLBCL	715, 980, 1122, 1258, 1398, 1449, 1587, 2033, 2260, 3868, 3984, 3991, 5086, 5124, 5578, 5670, 6015, 6153, 6191: **(19 Genes)**	22.6
Leukemia	87, 351, 557, 817, 925, 929, 1128, 1371, 1629, 2008, 2078, 2193, 2329, 4199, 4460, 4655, 4698, 4944: **(18 Genes)**	26.3
Bladder Cancer	700, 830, 1379, 2412, 2694, 2881, 3362, 3795, 3881, 3891, 3989, 4450, 4783, 4840, 5396, 5699: **(16 Genes)**	36.2
CNS	117, 293, 1430, 1770, 2051, 2353, 3127, 3696, 3946, 4547, 5254, 5812, 6179, 6789, 6962: **(15 Gens)**	42.1
ALL-AML	280, 461, 1177, 2692, 3021, 3137, 3208, 3404, 3426, 3504, 3596, 4951, 4955, 5049, 5468, 5986, 6116, 6472: **(18 Genes)**	26.3
Brain_Tumor2	592, 877, 1084, 3076, 3189, 3872, 3965, 3994, 4523, 4943, 5029, 5872, 6358, 6461, 7420, 7622, 7870, 9924, 9930, 10348: **(20 Genes)**	31.3
Prostate Tumor	1157, 1229, 1267, 1275, 1442, 1644, 1896, 2159, 3663, 4086, 4823, 5343, 5860, 6168, 6284, 6465, 6930, 7050, 8255, 9085, 9499: **(21 Genes)**	33.6
MLL	978, 2592, 3021, 4275, 5534, 7522, 7814, 8236, 8708, 8798, 9005, 9060, 9145, 10045, 10274, 10558, 10676, 12510: **(18 Genes)**	23.5
Breast Cancer	383, 541, 2802, 2881, 3669, 4263, 4279, 4731, 4915, 5182, 5312, 6690, 7369, 7796, 8775, 8785, 10164, 10888, 11903, 13063, 13556, 13620, 13700, 13799, 16611, 17334, 19416, 22286, 22508, 23776, 24117, 24427: **(32 Genes)**	42.8
COVID-19	1994, 2009, 4584, 4791, 5188, 5222, 6457, 7083, 7380, 8869, 11232, 11534, 12455, 12548, 12894, 13330, 14421, 16198, 17520, 17550, 17770, 18371, 18637, 18791, 18958, 19152, 19167, 19593, 19697, 20183, 20615, 20676, 21087, 23950, 24568, 24724: **(36 Genes)**	45.3

algorithms have delivered 100% accuracy in all datasets, except for Breast Cancer and COVID-19, in all runs. This outcome is considerably superior to the performance of all other divergent methodologies.

The results presented in Table 13 displays the quantity and mean number of chosen genes across various datasets runs by the BDE-BSS-DR algorithm. For example, the BDE-BSS-DR algorithm achieved 100% classification accuracy with 32 genes selected as effective in Breast Cancer from 24,481 genes. However, this algorithm selected 42.8 genes from this dataset on average. Comparably, the BDE-BSS-DR algorithms have obtained a classification accuracy of 100% for the COVID-19 dataset by selecting 36 genes (as recorded in Table 13). On the other hand, the SRBCT dataset has the smallest number of genes selected, which is only 11 genes. The proposed BDE-BSS-DR algorithm with SVM classifier and five-fold cross-validation is also examined on the COVID-19 dataset. The confusion matrix for the best-selected genes can be found in Appendix 6.

Fig. 5 also displays the convergence diagram of the BDE-BSS-DR. The convergence diagram of the BDE-BSS-DR algorithm presents important information about the algorithm search process. The figures show that despite the number of 500 repetitions for the BDE-BSS-DR in all datasets (except the two datasets Breast Cancer and COVID-19), the BDE-BSS-DR algorithm has achieved 100% classification accuracy in less than 300 iterations. In the other two datasets, Breast Cancer and COVID-19, the proposed method obtained better solutions in 500 iterations due to a large number of features. In certain datasets, such as SRBCT, DLBCL, Bladder Cancer, and MLL, the proposed method achieves 100% classification accuracy in less than 100 iterations (less than 3000 evaluations of the fitness function). Therefore, the DR mechanism is not used in the search process for these datasets. In some other datasets, such as Colon, Leukemia, Prostate Tumor, and MLL, the proposed method achieves 100% classification accuracy in less than 200 iterations (Less than 6000 evaluations of the fitness function). In two datasets, CNS and Brain_Tumor2, the algorithm has achieved 100% classification accuracy in less than 300 iterations.

Finally, as shown in Fig. 5, the algorithm needs more iterations in the breast and covid datasets to achieve 100% classification accuracy. Of course, observing the average classification accuracy of the algorithm in these datasets in Table 12 indicates that the BDE-BSS-DR algorithm in these datasets has not achieved 100% classification accuracy in all runs. Viewing the convergence diagram of the BDE-BSS-DR algorithm shows the high efficiency of the suggested BSS method as well as the DR mechanism in high-dimensional datasets.

4.4. Analysis of the DR mechanism and BSS algorithm

Many features in high-dimensional datasets create a very large search space. For example, in the COVID-19 dataset, which contains 25,534 features or genes, the number of solutions to the problem is $2^{25,534} - 1$. With wrapper-based FS algorithms in which a search algorithm (e.g., EC algorithms to search for optimal solutions) and a classification algorithm (e.g., SVM and KNN to evaluate the obtained solutions) are used, there are problems such as overfitting, the curse of dimensionality, high cost of computations, high memory usage and weak performance due to stopping at local optimal points. These problems cause the FS algorithm to not work properly. This was demonstrated in the performance of the BDE algorithm in Fig. 3, in high-dimensional datasets. To achieve high efficiency in FS in such datasets and reduce the problems expressed, the idea of this paper is to use a BSS and DR mechanism. BSS and DR mechanisms presented in this article can reduce the following problems:

- **Over-fitting problem:** High-dimensional data are often plagued by over-fitting due to the presence of numerous unimportant features. The DR mechanism is an effective approach to overcome this issue by physically eliminating these features. Moreover, the BSS algorithm can help to prevent overfitting by limiting the selection of a large number of features.
- **The curse of dimensionality:** In general, ML methods are designed for low-dimensional datasets. These methods do not work well in applications such as high-dimensional dataset classification. Therefore, using the BSS and DR mechanism can partially solve the problem of the curse of dimension.
- **The computational cost and high memory consumption:** Search agents represent problem solutions in wrapper-based FS algorithms. The large data size requires a lot of storage space to store search agents in memory. Moreover, the computational cost to evaluate the solutions obtained by the classifier and search algorithm in the search process is high. The increasing number of data samples exacerbates this problem. Therefore, the physical removal of dimensions can significantly reduce memory consumption and high computational costs. In addition, not selecting many features using the BSS algorithm can decrease the cost of computations of evaluating responses by the classifier method and performing calculations in the search algorithm.
- **Trap in local optima:** A vast problem space of high-dimensional data contains multiple local optimal points. However, the physical reduction of dimensions, resulting in a lower-dimensional space, causes a significant loss of such local optima. To mitigate this,

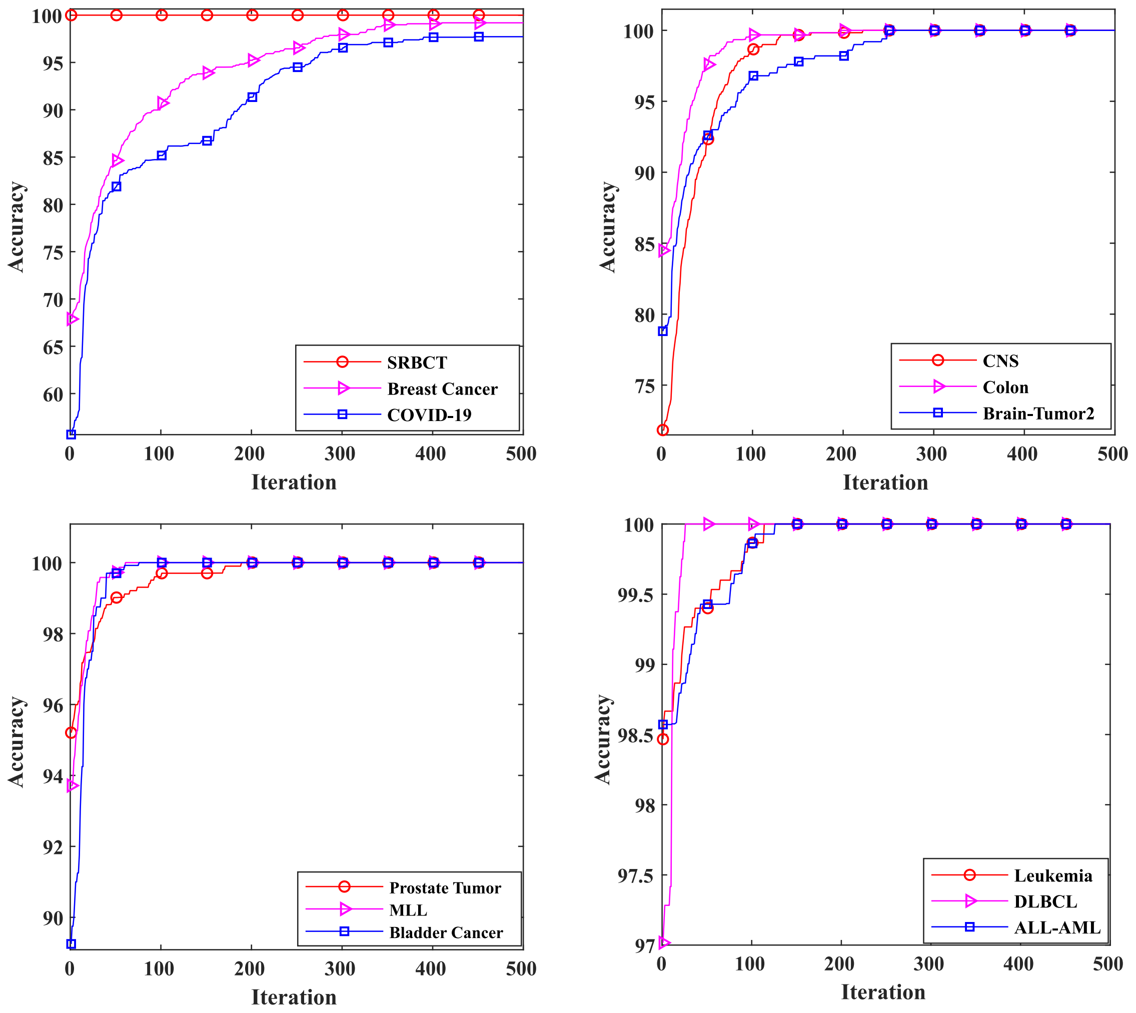


Fig. 5. The convergence diagram of the BDE-BSS-DR algorithm in gene selection application.

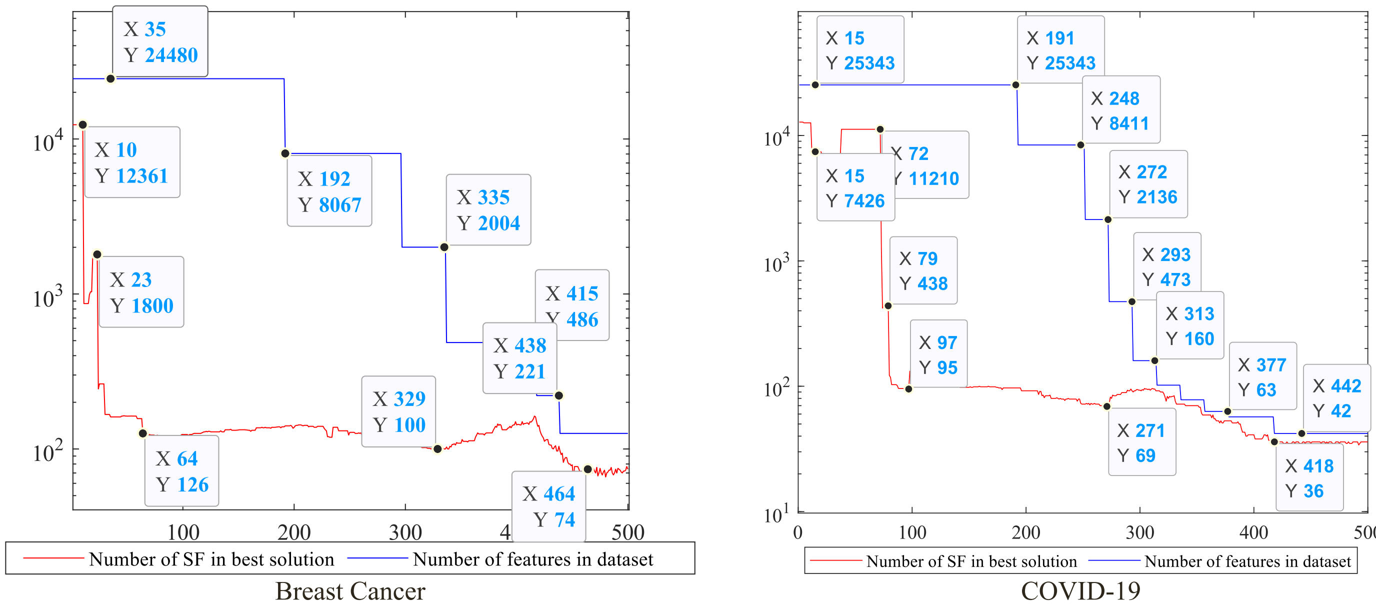


Fig. 6. The number of dataset genes and the number of genes selected by the best solution in the search process of the BDE-BSS-DR+SVM algorithm.

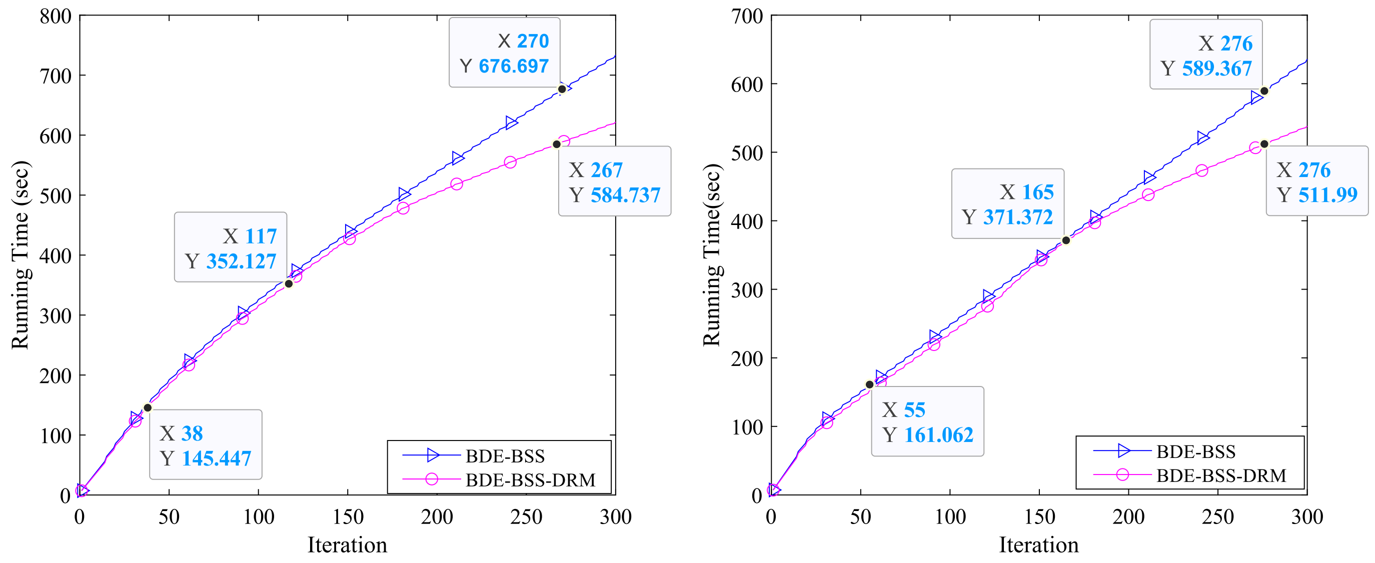


Fig. 7. The average running time(sec) of the BDE-BSS-DR and BDE-BSS on Breast Cancer and COVID-19.

employing BSS algorithms proves to be an effective approach for escaping from local optima.

Fig. 6 presents the behavior of the DR mechanism and the BSS algorithm in both COVID-19 and Breast Cancer datasets. As shown in Fig. 6, the DR mechanism in both datasets reduces the search space several times after 150 iterations. For example, in the COVID-19 dataset, up to 190 iterations, the number of features is 25,343. In this iteration, the DR mechanism is called, and it deleted 16,932 of the features. As a result, the algorithm continues the search process from iteration 191 to iteration 248 with 8411 features.

As we can see in Fig. 6, the last step of calling the DR mechanism is done in iteration number 418. In this way, the high-dimensional search space becomes a smaller-dimensional search space, and problems such as the curse of dimensionality, high memory consumption computational cost, and stopping at local optimal points are reduced. In addition to the behavior of the DR mechanism in Fig. 6, we see the number of features chosen by the best solution in different iterations. For example, up to 15 iterations, the search process continues with 12,678 features. In iteration 15, the BSS algorithm is called, and it achieves a better solution with 7426 genes by performing the search. Finally, the algorithm obtains the optimal solution with 36 genes in this run. We see the same process for the Breast Cancer dataset.

The mean running time of the suggested algorithms on the COVID-19 and Breast Cancer datasets was evaluated in 30 independent runs, as shown in Fig. 7, The outcomes of this evaluation were presented in Section 4.3.1. This comparison highlights the impact of the DR on the BDE-BSS-DR's running time. As depicted in the figures, both algorithms exhibit the same running time up to 150 iterations. However, in subsequent iterations, invoking the DR mechanism incurs a computational cost. Still, it reduces the number of remaining features in the search space by removing unimportant ones. This reduction in the number of features reduces the cost of computing the evaluation function and the algorithm compared to the BDE-BSS algorithm, and results in reducing the overall cost of the algorithm. Therefore, the DR mechanism not only improves classification accuracy and selects minimum features, but also reduces the computational cost in high-dimensional data.

The remarkable dominance of the BDE-BSS-DR and BDE-BSS over the BDE and other algorithms, especially in high-dimensional data, can be attributed to two factors. Firstly, the approach of the BSS algorithm is to consider the higher probability of not selecting features, which leads to a significant number of unimportant features being ignored in high-dimensional data. Secondly, the DR mechanism physically removes unimportant features from the search space, thereby reducing the computational cost of evaluating the fitness function, as explained earlier. An illustration in Fig. 6, provides a clear example of the BSS algorithm's efficacy. Within less than 100 iterations, the algorithm effectively changes a substantial number of less important features to non-selection mode, allowing relevant features to be added to the subset of features in subsequent iterations. This approach also reduces both computational cost and memory consumption. Furthermore, incorporating the DR mechanism and transforming the vast search space into a smaller search space during the search process leads to the elimination of numerous local optimal points, facilitating the discovery of better solutions. This approach also reduces both computational cost and memory consumption. Moreover, utilizing the BDE algorithm, particularly in the reduced search spaces, significantly contributes to enhancing the performance of the BDE-BSS-DR algorithm.

5. Conclusion and future recommendations

Efficient classification algorithms require the careful selection of relevant features and the removal of unimportant ones from the feature set. The medical field is an area where FS can be particularly impactful, as identifying significant features in medical data can help us better understand the factors that influence diseases. FS is also a crucial step in data preprocessing for designing ML-based medical decision support systems. Despite its benefits, FS can be inefficient, especially for high-dimensional data, due to problems such as the curse of dimensionality, local optima, cost of computations, and high memory consumption.

To overcome the issues, this paper suggested a novel FS method that uses the DR mechanism and BSS algorithm. The DR mechanism physically eliminated unimportant features identified during the search process, thereby reducing the search space. Gradual removal of features reduces the possibility of mistakenly removing relevant features before starting the search process. The BSS plays a vital role in improving the solutions obtained by the BDE algorithm. This algorithm, with a stochastic approach, plays an important role in escaping from the local optimal points and increasing the efficiency of the FS algorithms, particularly in high-dimensional data. The efficiency and effectiveness of each of the proposed algorithms were evaluated using various criteria such as classification accuracy, number of selected features and running time on 20 medical datasets. In addition, the importance and impact of each of the proposed methods, BSS and DR mechanism, as well as the initialization of search agents using the ReliefF algorithm, in increasing the efficiency of FS, were evaluated and analyzed separately. The results showed that each of the proposed methods has an important role in increasing the efficiency of FS, and their combination in the BDE-BSS-DR-r algorithm creates an efficient FS algorithm.

In future work, we plan to develop a framework for FS from high-dimensional datasets using EC methods, BSS methods, and the DR mechanism. In the proposed framework, an approach for self-organization of different parameters in the BSS algorithm and DR mechanism is provided. Also, to increase the convergence speed of the BSS algorithm, the best features determined by filter-based algorithms such as ReliefF and Fisher [62] can be used to add important features to the subset of selected features. Additionally, we will explore the use of enhanced EC methods instead of the BDE algorithm to further improve the efficiency and effectiveness of our BDE-BSS-DR and BDE-BSS algorithms, building on previous work that has notably increased FS efficiency for low-dimensional datasets (e.g., [63,64], and [65]).

CRediT authorship contribution statement

Karray Fakhri: Supervision, Writing – review & editing. **García Salvador:** Supervision, Writing – review & editing. **Acharya U Rajendra:** Supervision, Writing – review & editing. **Aghaei Leyla:** Formal analysis, Resources, Writing – original draft. **Safara Fatemeh:** Formal analysis, Resources, Writing – original draft. **Khosravi Abbas:** Supervision, Writing – review & editing. **Mirjalili Seyedali:** Supervision, Writing – review & editing. **Abdar Moloud:** Formal analysis, Methodology, Resources, Validation, Writing – review & editing, Supervision. **Ahadzadeh Behrouz:** Conceptualization, Data curation, Formal analysis, Methodology, Resources, Software, Writing – original draft.

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.

Data Availability

Data will be made available on request.

Appendix 1. Differential evolution (DE)

1.1. Generating initial population

The DE method starts by generating an initial population randomly within the problem space, which is characterized by a D-dimensional matrix denoted as D. The ith member of the population has the following structure:

$$X_{i,it} = \left(x_{1,i,it}, x_{2,i,it}, \ldots, x_{D,i,it}\right), i \in [1, \ldots, NP], \tag{1}$$

where $i \epsilon [1, \ldots, NP]$ represents the ith member of the population and $it = 1, \ldots, Max_{it}$ indicates the number of repetitions. The jth component of the ith member of the population is initialized as follows [43]:

$$x_{j,i,0} = x_{j,\min} + rand_{i,j}[0,1].\left(x_{j,\max} - x_{j,\min}\right), \tag{2}$$

where $rand_{i,j}[0,1]$ is a random number with a uniform distribution between 0 and 1, and $x_{j,\max}$ and $x_{j,\min}$ represent the upper and lower bounds of the jth dimension of the search space, respectively. After generating the initial population, the search process of the algorithm begins, and at each iteration, the mutation, crossover, and selection operators are applied to the search agents. The process continues until the optimal point is found [44].

1.2. Mutation operator

At each iteration, the DE algorithm creates a mutant vector $V_{i,it}$ for each target vector $X_{\mathrm{i,it}}$ in the population. Different mutation operators are proposed for the DE algorithm. One of the well-known mutation methods is presented in (3).

$$DE/rand/1 : V_{i,it} = X_{r1,i,it} + F.\left(X_{r2,i,it} - X_{r3,i,it}\right), \tag{3}$$

where $r1$, $r2$, and $r3$ are integers in the range of $[1, NP]$ that are selected randomly. Therefore, X_{r1}, X_{r2} and X_{r3} are population members selected randomly for the mutation procedure. The parameter F is called the scaling factor, which is a positive real number, and it is added to the vector difference X_{r2} and X_{r3} to amplify their difference [43]. Different values for F can affect the algorithm search process. For example, large values for F cause the global search algorithm to be performed in the entire problem space. Small values for F increase the algorithm's convergence speed, but the algorithm can be stopped at a local optimal point [42].

1.3. Crossover operator

The generated mutant vector $V_{i,it}$ using the mutation operation is combined with the target vector $X_{i,it}$ and a new solution $u_{i,it}$ called trial vector is generated as presented in Eq. (4) [44]:

$$u_{i,j,it} = \begin{cases} V_{i,j,it} & \text{if } rand \leq CR \text{ or } j = jrand \\ X_{i,j,it} & \text{otherwise} \end{cases} \quad (4)$$

At this stage, Eq. (4) reveals that the mutant vector $V_{i,it}$ components, accompanied by CR probability, are transmitted to the trial vector. Conversely, the components from the target vector $X_{i,it}$ are transmitted to the trial vector. In (4), *rand* is a random number in the range [0,1], and jrand denotes a number in the range of $[1, 2, \ldots, D]$ generated randomly. Also, CR is the crossover control parameter. Different values for CR affect the convergence of the algorithm. For instance, selecting a large value for the parameter CR increases the diversity of the population. The reason is that the trial vector inherits more information from the mutant vector $V_{i,it}$. In addition, using small values for the parameter CR causes a local search around the target vector $X_{i,it}$, because the trial vector $u_{i,it}$ changes are less than the changes of the target vector $X_{i,it}$, and the exploitation abilities of the algorithm increase slightly [43].

1.4. Selection operator

At this algorithm stage, some members of the population must be transferred to the next generation, and others must be eliminated. For this purpose, competition is made between the target vector $X_{i,it}$ and trial vector $u_{i,it}$ [42]. The selection operator for this algorithm is performed according to Eq. (5):

$$X_{i,it+1} = \begin{cases} u_{i,it} & \text{if } fit(u_{i,it}) \geq fit(X_{i,it}) \\ X_{i,it} & \text{otherwise} \end{cases} \quad (5)$$

where $fit(X_{i,it})$ is the fitness function to be maximized. If the $fit(u_{i,it})$ is greater than or equal to $fit(X_{i,it})$, then trial vector $u_{i,it}$ is transferred to the next generation, and otherwise target vector $X_{i,it}$ is transferred to the next generation [42].

Appendix 2

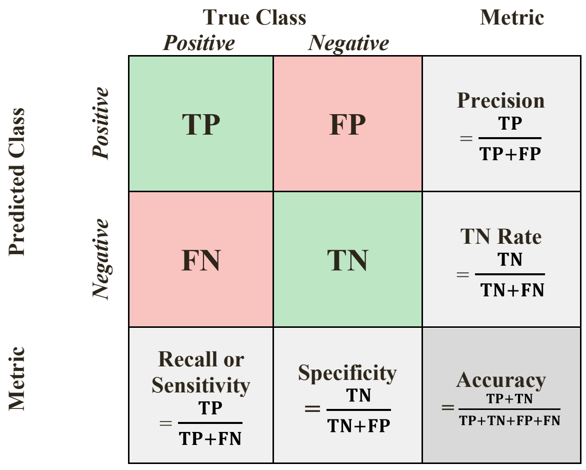


Fig. 1. Confusion matrix and evaluation metrics.

Appendix 3

1. https://data.mendeley.com/datasets/fhx5zgx2zj/1
2. http://archive.ics.uci.edu/ML/index.php
3. https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE149273
4. https://schlieplab.org/Static/Supplements/CompCancer/datasets.htm.

Appendix 4

A

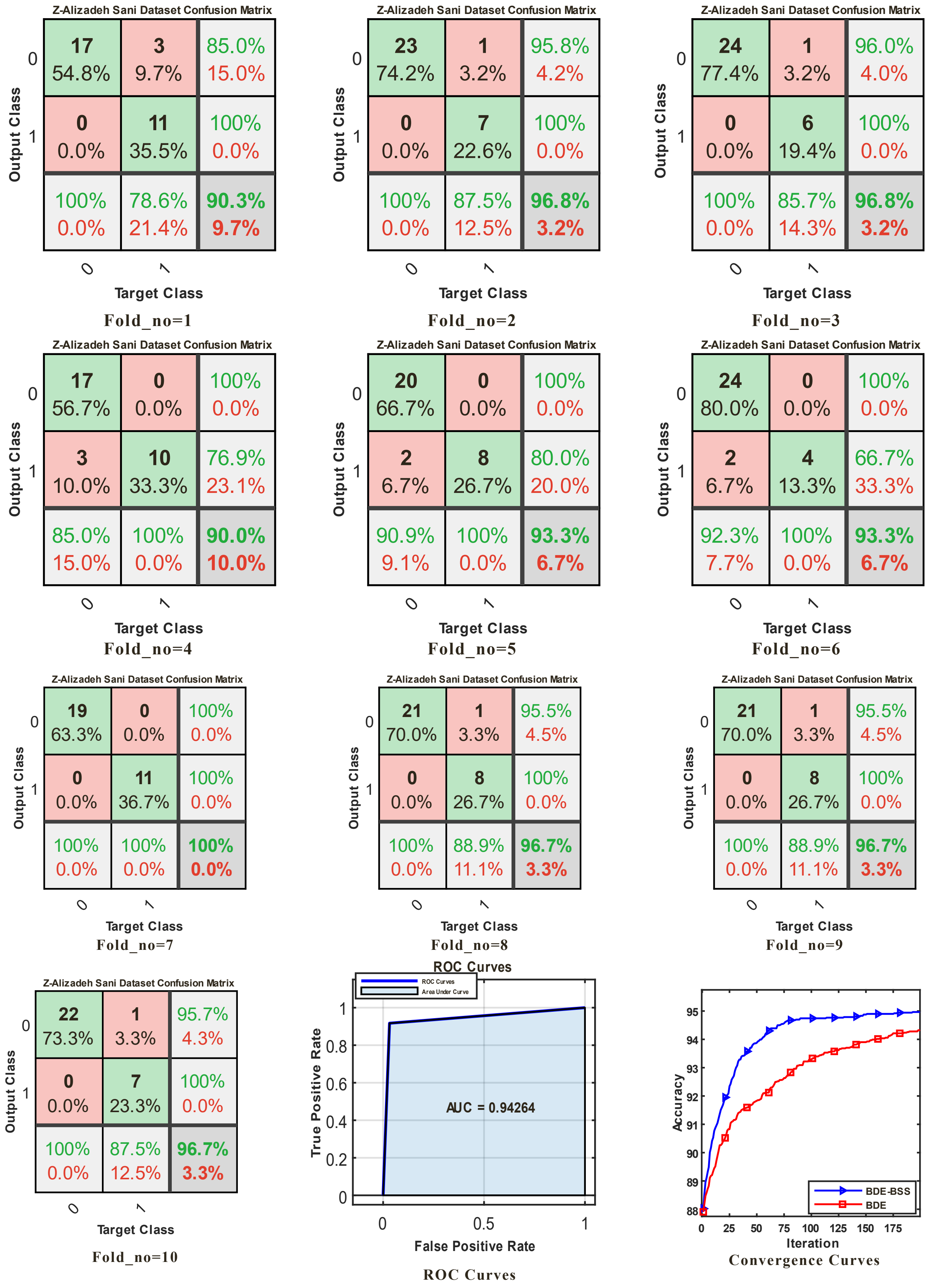


Fig. 2. The proposed method confusion matrix, with the 20 features selected by BDE-BSS+SVM in different folds of 10CV.

.

Appendix 5

Table 1
The Z-Alizadeh Sani dataset with all features and the status of features after FS using the BDE-BSS algorithm.

No.	Feature name	Value	SF	No.	Feature name	Value	SF
1	Age	[30–86]	**1**	28	Nonanginal chest pain	yes, no	0
2	Weight	[48–120]	**1**	29	Exertional chest pain	yes, no	**1**
3	Sex	F, M	0	30	Low Th Ang (low-Threshold angina)	yes, no	0
4	BMI (Body Mass Index Kg/m^2)	18–41	**1**	31	Rhythm	Sin, AF	0
5	DM (Diabetes Mellitus)	yes, no	0	32	Q wave	yes, no	**1**
6	HTN (Hypertension)	yes, no	**1**	33	ST elevation	yes, no	0
7	Current smoker	yes, no	**1**	34	ST depression	yes, no	**1**
8	Ex-smoker	yes, no	0	35	T inversion	yes, no	**1**
9	FH (Family History)	yes, no	0	36	LVH (Left Ventricular Hypertrophy)	yes, no	**0**
10	Obesity	if BMI> 25 yes, else no	0	37	Poor R-wave progression	yes, no	**1**
11	CRF (Chronic Renal Failure)	yes, no	0	38	FBS (Fasting Blood Sugar mg/dL)	[62–400]	0
12	CVA (Cerebrovascular Accident)	yes, no	0	39	Cr (Creatine mg/dL)	[0.5–2.2]	0
13	Airway disease	yes, no	0	40	TG (Triglyceride mg/dL)	[37–1050]	0
14	Thyroid disease	yes, no	0	41	LDL (Low-Density Lipoprotein mg/dL)	[18–232]	0
15	CHF (Congestive Heart Failure)	yes, no	**1**	42	HDL (High-Density Lipoprotein mg/Dl)	[15–111]	0
16	DLP (Dyslipidemia)	yes, no	0	43	BUN (Blood Urea Nitrogen mg/dL)	[6–52]	0
17	BP (Blood Pressure mm Hg)	90–190	**1**	44	ESR (Erythrocyte Sedimentation Rate mm/h)	[1–90]	**1**
18	PR (Pulse Rate ppm)	50–110	0	45	HB (Hemoglobin g/dL)	[8.9–17.6]	**1**
19	Edema	yes, no	0	46	K (Potassium mEq/lit)	[3.0–6.6]	**1**
20	Weak peripheral pulse	yes, no	0	47	Na (Sodium mEq/lit)	[128–156]	0
21	yes, no	yes, no	0	48	WBC (White Blood Cell cells/ML)	[3700–18,000	0
22	Systolic murmur	yes, no	0	49	Lymph (Lymphocyte %)	[7–60]	0
23	Diastolic murmur	yes, no	**1**	50	Neut (Neutrophil %)	[32–89]	0
24	Typical chest pain	yes, no	**1**	51	PLT (Platelet 1000/ML)	[25–742]	**1**
25	Dyspnea	yes, no	0	52	EF (Ejection Fraction %)	[15–60]	**1**
26	Function class	1, 2, 3, 4	0	53	Region with RWMA	0,1,2,3,4	**1**
27	Atypical	yes, no	0	54	VHD (Valvular Heart Disease)	Normal, Mild, Moderate, Severe	0

Note: SF is the status of features after FS using the BDE-BSS algorithm: 1 means that the feature is selected, and 0 means that the feature is not selected.

Appendix 6

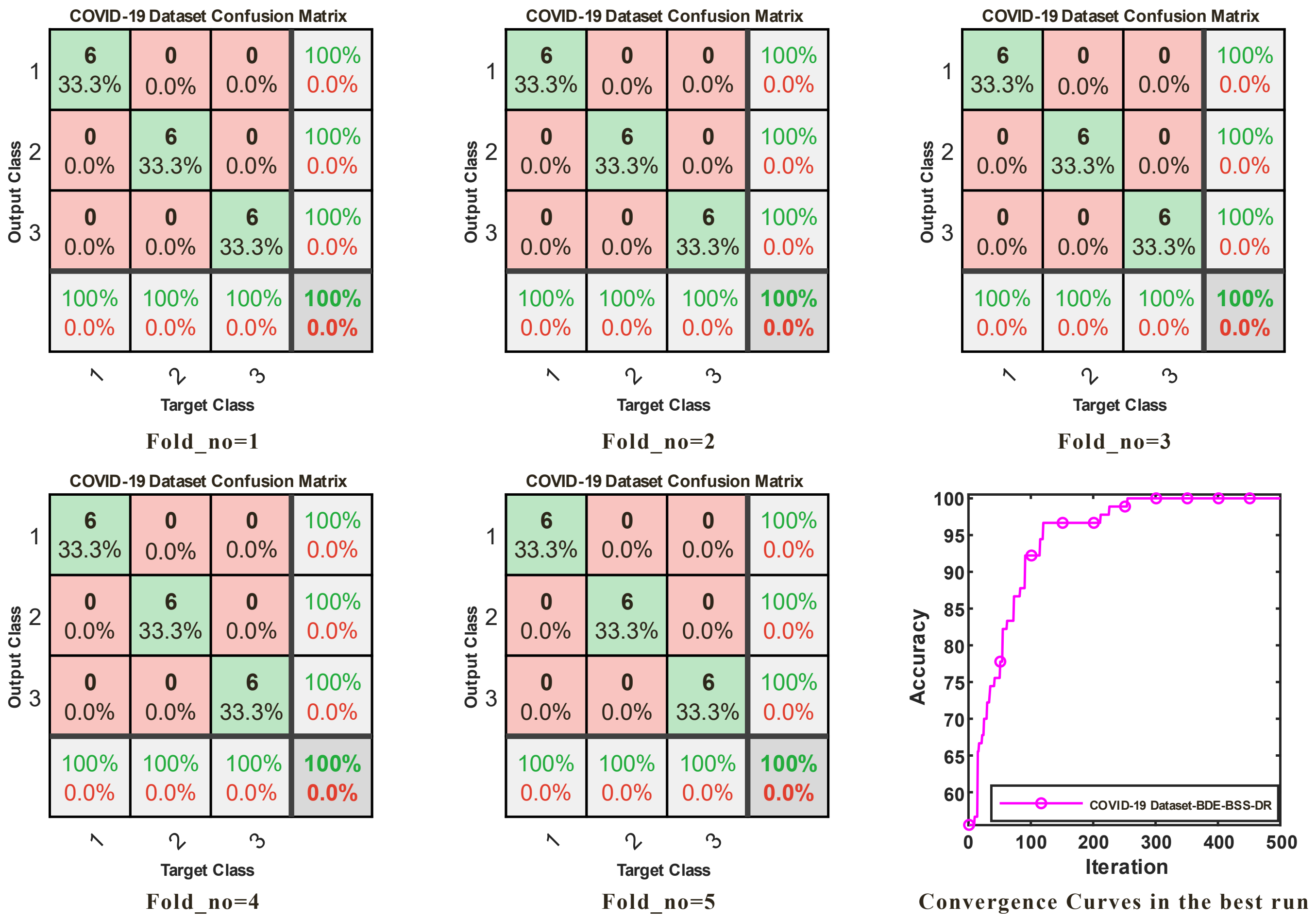


Fig. 3. Confusion matrix by the proposed method using 36 genes selected by BDE-BSS-DR+SVM in different folds of 5CV.

.

References

[1] P. Moradi, M. Gholampour, A hybrid particle swarm optimization for feature subset selection by integrating a novel local search strategy, Appl. Soft Comput. 43 (2016) 117–130.
[2] B. Xue, M. Zhang, W.N. Browne, X. Yao, A survey on evolutionary computation approaches to feature selection, IEEE Trans. Evolut. Comput. 20 (4) (2015) 606–626.
[3] B. Ahadzadeh, M. Abdar, F. Safara, A. Khosravi, M.B. Menhaj, P.N. Suganthan, SFE: a simple, fast and efficient feature selection algorithm for high-dimensional data, IEEE Trans. Evolut. Comput. (2023).
[4] B. Remeseiro, V. Bolon-Canedo, A review of feature selection methods in medical applications, Comput. Biol. Med. 112 (2019), 103375.
[5] Y. Xue, T. Tang, W. Pang, A.X. Liu, Self-adaptive parameter and strategy-based particle swarm optimization for large-scale feature selection problems with multiple classifiers, Appl. Soft Comput. 88 (2020), 106031.
[6] T. Li, Z.H. Zhan, J.C. Xu, Q. Yang, Y.Y. Ma, A binary individual search strategy-based bi-objective evolutionary algorithm for high-dimensional feature selection, Inf. Sci. 610 (2022) 651–673.
[7] M. Li, L. Ke, L. Wang, S. Deng, X. Yu, A novel hybrid gene selection for tumor identification by combining multifilter integration and a recursive flower pollination search algorithm, Knowl. -Based Syst. 262 (2023), 110250.
[8] K. Chen, B. Xue, M. Zhang, F. Zhou, Evolutionary multitasking for feature selection in high-dimensional classification via particle swarm optimization, IEEE Trans. Evolut. Comput. 26 (3) (2021) 446–460.
[9] Y. Zhu, W. Li, T. Li, A hybrid Artificial Immune optimization for high-dimensional feature selection, Knowl. -Based Syst. 260 (2023), 110111.
[10] M. Robnik-Šikonja, I. Kononenko, Theoretical and empirical analysis of ReliefF and RReliefF, Mach. Learn. 53 (1) (2003) 23–69.
[11] H. Saadatmand, M.R. Akbarzadeh-T, Set-based integer-coded fuzzy granular evolutionary algorithms for high-dimensional feature selection, Appl. Soft Comput. 142 (2023), 110240.
[12] W. Wei, M. Xuan, L. Li, Q. Lin, Z. Ming, C.A.C. Coello, Multiobjective optimization algorithm with dynamic operator selection for feature selection in high-dimensional classification, Appl. Soft Comput. 143 (2023), 110360.
[13] X. Wang, Y. Wang, K.C. Wong, X. Li, A self-adaptive weighted differential evolution approach for large-scale feature selection, Knowl. -Based Syst. 235 (2022), 107633.
[14] O. Tarkhaneh, T.T. Nguyen, S. Mazaheri, A novel wrapper-based feature subset selection method using modified binary differential evolution algorithm, Inf. Sci. 565 (2021) 278–305.
[15] J. Zhou, Z. Hua, A correlation guided genetic algorithm and its application to feature selection, Appl. Soft Comput. 123 (2022), 108964.
[16] X. Li, Q. Fu, Q. Li, W. Ding, F. Lin, Z. Zheng, Multi-objective binary grey wolf optimization for feature selection based on guided mutation strategy, Appl. Soft Comput. (2023), 110558.
[17] E. Zorarpacı, S.A. Özel, A hybrid approach of differential evolution and artificial bee colony for feature selection, Expert Syst. Appl. 62 (2016) 91–103.
[18] M. Mafarja, I. Aljarah, H. Faris, A.I. Hammouri, A.Z. Ala'M, S. Mirjalili, Binary grasshopper optimisation algorithm approaches for feature selection problems, Expert Syst. Appl. 117 (2019) 267–286.
[19] E.S.M. El-Kenawy, M.M. Eid, M. Saber, A. Ibrahim, MbGWO-SFS: Modified binary grey wolf optimizer based on stochastic fractal search for feature selection, IEEE Access 8 (2020) 107635–107649.
[20] Y. Zhang, D.W. Gong, X.Z. Gao, T. Tian, X.Y. Sun, Binary differential evolution with self-learning for multi-objective feature selection, Inf. Sci. 507 (2020) 67–85.
[21] M. Tubishat, S. Ja'afar, M. Alswaitti, S. Mirjalili, N. Idris, M.A. Ismail, M.S. Omar, Dynamic salp swarm algorithm for feature selection, Expert Syst. Appl. 164 (2021), 113873.

[22] P. Tan, X. Wang, Y. Wang, Dimensionality reduction in evolutionary algorithms-based feature selection for motor imagery brain-computer interface, Swarm Evolut. Comput. 52 (2020), 100597.
[23] F. Cheng, F. Chu, Y. Xu, L. Zhang, A steering-matrix-based multiobjective evolutionary algorithm for high-dimensional feature selection, IEEE Trans. Cybern. (2021).
[24] X.F. Song, Y. Zhang, D.W. Gong, X.Y. Sun, Feature selection using bare-bones particle swarm optimization with mutual information, Pattern Recognit. 112 (2021), 107804.
[25] F. Kılıç, Y. Kaya, S. Yildirim, A novel multi population-based particle swarm optimization for feature selection, Knowl. -Based Syst. 219 (2021), 106894.
[26] X.F. Song, Y. Zhang, Y.N. Guo, X.Y. Sun, Y.L. Wang, Variable-size cooperative coevolutionary particle swarm optimization for feature selection on high-dimensional data, IEEE Trans. Evolut. Comput. 24 (5) (2020) 882–895.
[27] L. Li, M. Xuan, Q. Lin, M. Jiang, Z. Ming, K.C. Tan, An evolutionary multitasking algorithm with multiple filtering for high-dimensional feature selection, IEEE Trans. Evolut. Comput. (2023).
[28] H. Pan, S. Chen, H. Xiong, A high-dimensional feature selection method based on modified Gray Wolf Optimization, Appl. Soft Comput. 135 (2023), 110031.
[29] L. Qu, W. He, J. Li, H. Zhang, C. Yang, B. Xie, Explicit and size-adaptive PSO-based feature selection for classification, Swarm Evolut. Comput. 77 (2023), 101249.
[30] F. Cheng, J. Cui, Q. Wang, L. Zhang, A variable granularity search-based multiobjective feature selection algorithm for high-dimensional data classification, IEEE Trans. Evolut. Comput. 27 (2) (2022) 266–280.
[31] Chen, K., Xue, B., Zhang, M., & Zhou, F. (2020). An evolutionary multitasking-based feature selection method for high-dimensional classification. *IEEE Transactions on Cybernetics.*
[32] L. Sun, S. Si, W. Ding, X. Wang, J. Xu, Multiobjective sparrow search feature selection with sparrow ranking and preference information and its applications for high-dimensional data, Appl. Soft Comput. 147 (2023), 110837.
[33] M. Zhang, J.S. Wang, Y. Liu, H.M. Song, J.N. Hou, Y.C. Wang, M. Wang, Multi-objective optimization algorithm based on clustering guided binary equilibrium optimizer and NSGA-III to solve high-dimensional feature selection problem, Inf. Sci. 648 (2023), 119638.
[34] Z. Li, A local opposition-learning golden-sine grey wolf optimization algorithm for feature selection in data classification, Appl. Soft Comput. 142 (2023), 110319.
[35] S. Deng, Y. Li, J. Wang, R. Cao, M. Li, A feature-thresholds guided genetic algorithm based on a multi-objective feature scoring method for high-dimensional feature selection, Appl. Soft Comput. 148 (2023), 110765.
[36] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evolut. Comput. 1 (1) (1997) 67–82.
[37] Wang, P., Xue, B., Liang, J., & Zhang, M. (2022). Differential Evolution with Duplication Analysis for Feature Selection in Classification. IEEE Transactions on Cybernetics.
[38] Wang, P., Xue, B., Liang, J., & Zhang, M. (2023). Feature Selection Using Diversity-Based Multi-objective Binary Differential Evolution. *Information Sciences*.
[39] Wang, P., Xue, B., Liang, J., & Zhang, M. (2022). Differential evolution-based feature selection: A niching-based multi-objective approach. *IEEE Transactions on Evolutionary Computation.*
[40] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, J. Glob. Optim. 11 (4) (1997) 341–359.
[41] J. Cheng, Z. Pan, H. Liang, Z. Gao, J. Gao, Differential evolution algorithm with fitness and diversity ranking-based mutation operator, Swarm Evolut. Comput. 61 (2021), 100816.
[42] Y. Wang, H.X. Li, T. Huang, L. Li, Differential evolution based on covariance matrix learning and bimodal distribution parameter setting, Appl. Soft Comput. 18 (2014) 232–247.
[43] Y. Wang, Z. Cai, Q. Zhang, Differential evolution with composite trial vector generation strategies and control parameters, IEEE Trans. Evolut. Comput. 15 (1) (2011) 55–66.
[44] S. Das, P.N. Suganthan, Differential evolution: a survey of the state-of-the-art, IEEE Trans. Evolut. Comput. 15 (1) (2010) 4–31.
[45] S. Das, S.S. Mullick, P.N. Suganthan, Recent advances in differential evolution–an updated survey, Swarm Evolut. Comput. 27 (2016) 1–30.
[46] L. Wang, X. Fu, Y. Mao, M.I. Menhas, M. Fei, A novel modified binary differential evolution algorithm and its applications, Neurocomputing 98 (2012) 55–75.
[47] E. Nasarian, M. Abdar, M.A. Fahami, R. Alizadehsani, S. Hussain, M.E. Basiri, N. Sarrafzadegan, Association between work-related features and coronary artery disease: a heterogeneous hybrid feature selection integrated with balancing approach, Pattern Recognit. Lett. 133 (2020) 33–40.
[48] M. Abdar, W. Książek, U.R. Acharya, R.S. Tan, V. Makarenkov, P. Pławiak, A new machine learning technique for an accurate diagnosis of coronary artery disease, Comput. Methods Prog. Biomed. 179 (2019), 104992.
[49] R. Alizadehsani, J. Habibi, M.J. Hosseini, H. Mashayekhi, R. Boghrati, A. Ghandeharioun, Z.A. Sani, A data mining approach for diagnosis of coronary artery disease, Comput. Methods Prog. Biomed. 111 (1) (2013) 52–61.
[50] Z. Arabasadi, R. Alizadehsani, M. Roshanzamir, H. Moosaei, A.A. Yarifard, Computer aided decision making for heart disease detection using hybrid neural network-Genetic algorithm, Comput. Methods Prog. Biomed. 141 (2017) 19–26.
[51] C.J. Qin, Q. Guan, X.P. Wang, Application of ensemble algorithm integrating multiple criteria feature selection in coronary heart disease detection, Biomed. Eng. Appl. Basis Commun. 29 (06) (2017) 1750043.
[52] F. Babič, J. Olejár, Z. Vantová, J. Paralič, Predictive and descriptive analysis for heart disease diagnosis (September). *In 2017 federated conference on computer science and information systems* (fedcsis), IEEE, 2017, pp. 155–163 (September).
[53] M. Abdar, U.R. Acharya, N. Sarrafzadegan, V. Makarenkov, NE-nu-SVC: a new nested ensemble clinical decision support system for effective diagnosis of coronary artery disease, IEEE Access 7 (2019) 167605–167620.
[54] B.A. Tama, S. Im, S. Lee, Improving an intelligent detection system for coronary heart disease using a two-tier classifier ensemble, BioMed. Res. Int. (2020) 2020.
[55] M.M. Ghiasi, S. Zendehboudi, A.A. Mohsenipour, Decision tree-based diagnosis of coronary artery disease: CART model, Comput. Methods Prog. Biomed. 192 (2020), 105400.
[56] A.H. Shahid, M.P. Singh, A novel approach for coronary artery disease diagnosis using hybrid particle swarm optimization based emotional neural network, Biocybern. Biomed. Eng. 40 (4) (2020) 1568–1585.
[57] Ashish, L., Kumar, S., & Yeligeti, S. (2021). Ischemic heart disease detection using support vector machine and extreme gradient boosting method. *Materials Today: Proceedings.*
[58] E. Pashaei, E. Pashaei, Gene selection using hybrid dragonfly black hole algorithm: a case study on RNA-seq COVID-19 data, Anal. Biochem. 627 (2021), 114242.
[59] O.A. Alomari, S.N. Makhadmeh, M.A. Al-Betar, Z.A.A. Alyasseri, I.A. Doush, A. K. Abasi, R.A. Zitar, Gene selection for microarray data classification based on Gray Wolf Optimizer enhanced with TRIZ-inspired operators, Knowl. -Based Syst. 223 (2021), 107034.
[60] A. Dabba, A. Tari, S. Meftali, R. Mokhtari, Gene selection and classification of microarray data method based on mutual information and moth flame algorithm, Expert Syst. Appl. 166 (2021), 114012.
[61] A. Dabba, A. Tari, S. Meftali, Hybridization of Moth flame optimization algorithm and quantum computing for gene selection in microarray data, J. Ambient Intell. Humaniz. Comput. 12 (2) (2021) 2731–2750.
[62] Gu, Q., Li, Z., & Han, J. (2012). Generalized fisher score for feature selection. *arXiv preprint arXiv:1202.3725*.
[63] B. Turkoglu, S.A. Uymaz, E. Kaya, Binary artificial algae algorithm for feature selection, Appl. Soft Comput. 120 (2022), 108630.
[64] X.H. Wang, Y. Zhang, X.Y. Sun, Y.L. Wang, C.H. Du, Multi-objective feature selection based on artificial bee colony: an acceleration approach with variable sample size, Appl. Soft Comput. 88 (2020), 106041.
[65] Wang, Z., Gao, S., Zhou, M., Sato, S., Cheng, J., & Wang, J. (2022). Information-theory-based nondominated sorting ant colony optimization for multiobjective feature selection in classification. *IEEE Transactions on Cybernetics*.