2020

Cooperative co-evolution for feature selection in big data with random feature grouping

A.N.M. Bazlur Rashid
Edith Cowan University

Mohiuddin Ahmed
Edith Cowan University

Leslie F. Sikos
Edith Cowan University

Paul Haskell-Dowland
Edith Cowan University

10.1186/s40537-020-00381-y
Rashid, A. N. M. B., Ahmed, M., Sikos, L. F., & Haskell-Dowland, P. (2020). Cooperative co-evolution for feature selection in big data with random feature grouping. *Journal of Big Data, 7*, article 107. https://doi.org/10.1186/s40537-020-00381-y

Rashid *et al. J Big Data* (2020) 7:107
https://doi.org/10.1186/s40537-020-00381-y

Journal of Big Data

RESEARCH Open Access

Cooperative co-evolution for feature selection in Big Data with random feature grouping

A. N. M. Bazlur Rashid*, Mohiuddin Ahmed, Leslie F. Sikos and Paul Haskell-Dowland

*Correspondence:
a.rashid@ecu.edu.au
School of Science, Edith Cowan University, Joondalup, WA, Australia

Abstract

A massive amount of data is generated with the evolution of modern technologies. This high-throughput data generation results in Big Data, which consist of many features (attributes). However, irrelevant features may degrade the classification performance of machine learning (ML) algorithms. Feature selection (FS) is a technique used to select a subset of relevant features that represent the dataset. Evolutionary algorithms (EAs) are widely used search strategies in this domain. A variant of EAs, called cooperative co-evolution (CC), which uses a divide-and-conquer approach, is a good choice for optimization problems. The existing solutions have poor performance because of some limitations, such as not considering feature interactions, dealing with only an even number of features, and decomposing the dataset statically. In this paper, a novel random feature grouping (RFG) has been introduced with its three variants to dynamically decompose Big Data datasets and to ensure the probability of grouping interacting features into the same subcomponent. RFG can be used in CC-based FS processes, hence called *Cooperative Co-Evolutionary-Based Feature Selection with Random Feature Grouping (CCFSRFG)*. Experiment analysis was performed using six widely used ML classifiers on seven different datasets from the UCI ML repository and Princeton University Genomics repository with and without FS. The experimental results indicate that in most cases [i.e., with naïve Bayes (NB), support vector machine (SVM), *k*-Nearest Neighbor (*k*-NN), J48, and random forest (RF)] the proposed CCFSRFG-1 outperforms an existing solution (a CC-based FS, called CCEAFS) and CCFSRFG-2, and also when using all features in terms of accuracy, sensitivity, and specificity.

Keywords: Big Data, Feature selection, Cooperative co-evolution, Random feature grouping, Machine learning

Introduction

The generation of massive volumes of data in the *Big Data* era is common in many areas, including, but not limited to, the Internet of Things (IoT), cybersecurity, and healthcare [1]. Big Data opens the door to the research community for explore new knowledge, and often *machine learning (ML)* algorithms are used to learn, predict, and classify data in this context. The use of ML classifiers in different application domains, for example, healthcare and cybersecurity, have been studied in the literature [2]. Real-world

problems across various domains consist of several features (attributes in dataset terminology). However, not all of these features are important, because some are irrelevant or redundant, and as such, may degrade the performance of ML classifiers [3, 4]. *Feature selection (FS)* is an approach to select the relevant features for reducing the dimension of data in order to improve ML performance [5]. Formally speaking, feature selection is a process to select a subset of s features from a full set of n features ($s < n$) in a dataset by removing irrelevant and unimportant features, thereby representing it with less features [2]. A search technique initiates the FS process to discover feature subsets. Then, feature subsets are evaluated by different performance measures, for example, classification accuracy. A terminating criterion, for instance, the maximum number of generations, is used to terminate the FS process. A validation method at the end of the FS process can test the validity of the selected subset of features.

A dataset consisting of n features has 2^k possible solutions, which makes the feature selection process computationally expensive. A wide range of search techniques can be applied to the FS process, for example, greedy search, best search, or evolutionary search [6]. *Evolutionary algorithms (EA)* are search techniques that are widely used for feature selection [7]. However, with the increase in data samples and features in a dataset, the search space also increases. In most cases, this impacts the effectiveness of EAs. *Cooperative co-evolution (CC)*, a meta-heuristic algorithm, follows a divide-and-conquer technique and has been effectively applied in different domains, including a limited number of FS applications. CC decomposes each complex problem into multiple subproblems, optimizes each subproblem individually, and collaborates different subproblems to build a complete solution to the problem [5]. Hence, a CC-based FS approach with a suitable decomposition method, an appropriate optimizer, and a proper collaboration technique can be studied for feature selection problems in different Big Data domains. However, the performance of a CC algorithm (i.e., solution quality) depends on how the problem is decomposed [8–12]. The performance of a CC algorithm may degrade significantly if the problem is *non-separable*, such as if the features interact with each other [9, 13, 14]. In the literature, it is suggested that the interdependency between the decomposed subproblems should be minimized because of the collaboration requirement of CC algorithms [15]. However, for real-world problems without any prior information about how the features in a dataset interact, it is difficult to find a suitable problem decomposition technique for feature selection. Non-CC-based feature grouping methods are also investigated in the literature [16–18]. This paper aims to investigate CC-based feature grouping.

To overcome the aforementioned limitations, this paper introduces a CC-based FS framework with a proposed *random feature grouping (RFG)* decomposition technique: *Cooperative Co-Evolutionary-Based Feature Selection with Random Feature Grouping (CCFSRFG)*. This approach has been evaluated with two variants using six widely used ML classifiers on seven different Big Data datasets from the *UCI ML repository*[1] and Princeton University Genomics Repository.[2] Based on the analysis of the comparative results, it has been observed that in terms of accuracy, sensitivity, and specificity,

[1] http://archive.ics.uci.edu/ml/.

[2] https://www.princeton.edu/.

Table 1 Taxonomy of FS approaches [5, 74, 75]

FS approaches	Evaluation methods	Examples of evaluator
Evaluation criteria	Filter [76]	Distributed FS using SC [77], information theory [78], *T*-test [79]
	Wrapper [80]	*k*-NN [81], SVM [82]
	Embedded [83]	LASSO [84], Gradient boosting [85]
Evolutionary computation	EA [86]	GA [87], GP [88], parallel GA [89]
	CEA [25]	CCEA [90]
	Swarm optimization [87]	PSO [87], ACO [91]
	Hybrid	mRMR-TLBOL [92], CMIM + BGA [93], TLBO + GSA [94]
	Others	ABC [95], MA [96], DE [95]
Number of objectives	Single-objective [97]	GA [87]
	Multi-objectives [98]	Nondominated sorting GA-II [99]

GP genetic programming, *PSO* particle swarm optimization, *ACO* ant colony optimization, *TLBO* teaching learning-based algorithm, *GSA* gravitational search algorithm, *CMIM* conditional mutual information maximization, *BGA* binary genetic algorithm, *mRMR* minimum redundancy maximum relevance, *TLBOL* TLBO with opposition-based learning, *DE* differential evolution, *MA* memetic algorithm, *LCS* learning classifier system, *ES* evolutionary strategy, *ABC* artificial bee colony

CCFSRFG with RFG-1 in most cases outperforms CCEAFS [19] and the classifiers when using all features, along with a significant feature reduction.

The rest of the paper is organized as follows. Next section presents feature selection approaches using a taxonomy and an extensive literature review of problem decomposition methods. After that, a novel cooperative co-evolution technique and a novel feature selection framework is included. Then, a proposed random feature grouping with three variants is illustrated. The experimental results are presented and analyzed in "Results and discussions" section. The conclusion and future work directions are included in the last section.

Literature review

Many feature selection approaches studied in the literature are based on various metrics, such as information theory, probability distribution, and classification accuracy [20]. Table 1 presents a taxonomy of these.

There are a few FS research works that have been studied in the literature based on CC [21–29]. The literature indicates that the FS based on CC is an emerging research field and it is yet to be investigated for optimization problems. Based on this finding, a CC-based FS framework (CCEAFS) has been proposed in our previous study [19]. In CCEAFS, the FS problem has been decomposed into subproblems based on static decomposition; each of these subproblems has been optimized using a genetic algorithm, and a complete solution to the problem has been created using $N + 1$ collaboration with random collaboration for the first generation, and using best individuals as collaborators for further generations. CCEAFS has a promising performance and significant feature reduction capabilities while maintaining classification performance, and in a few cases, it performs better than using all features. Nevertheless, CCEAFS has some limitations: (1) it does not consider feature interactions for features with tightly coupled relationships and which potentially increase classification accuracy; (2) it considers datasets with an even number of features and decomposed

datasets statically with an even number of features in each subdataset; however, in practice, datasets may have an odd number of features; (3) it considers datasets with a high number of samples and a low number of features and has not been validated using datasets with other characteristics. The performance of a CC mostly depends on the appropriate decomposition methods, suitable optimizers, and proper collaboration techniques [30]. This paper aims to overcome the limitations of the previously studied CCEAFS framework for feature selection in Big Data, and also to propose a new decomposition method for feature selection.

Preliminaries

This section defines the variable interaction and problem structure, whether it is separable, partially-separable, partially additively separable, or non-separable.

Definition 1 Decision variables i and j from a decision vector x are *interacting* if the ith and jth variables can be replaced by x_i' and x_j' such that [31]:

$$\begin{aligned}&\left(f\left(x_1,\ldots,x_i,\ldots,x_j,\ldots,x_n\right) < f\left(x_1,\ldots,x_i',\ldots,x_j,\ldots,x_n\right)\right)\wedge\\&\left(f\left(x_1,\ldots,x_i,\ldots,x_j',\ldots,x_n\right) > f\left(x_1,\ldots,x_i',\ldots,x_j',\ldots,x_n\right)\right)\end{aligned} \tag{1}$$

where $x = \{x_1, x_2, \ldots, x_n\}$ is a decision vector of n decision variables.

Definition 2 Function $f(x)$ of n decision variables is called *separable* if it is represented as a sum of n independent functions having only one independent variable in each [32, 33]:

$$\underset{x_1,x_2,\ldots,x_n}{\operatorname{argmin}} f(x_1,x_2,\ldots,x_n) = \left\{\underset{x_1}{\operatorname{argmin}} f(x_1,\ldots), \underset{x_2}{\operatorname{argmin}} f(x_2,\ldots),\ldots,\underset{x_n}{\operatorname{argmin}} f(\ldots,x_n)\right\}. \tag{2}$$

The Rastrigin function is an example of separable functions [34]:

$$f(x) = 3.0n + \sum_{i=1}^{n} x_i^2 - 3.0\ cos(2\pi x_i), \tag{3}$$

where $-5.12 \leqslant x_i \leqslant 5.12$ and the global minimum of (3) is at the point $x = \{0, 0, 0, \ldots, 0\}$. One of the characteristics of this function is that it has many suboptimal peaks whose values are increased with the increase of the distance from the global optimal point.

Definition 3 Function $f(x)$ of n decision variables is called *partially separable* having m independent subcomponents of the form [35]:

$$\underset{x_1,x_2,\ldots,x_n}{\operatorname{argmin}} f(x_1,x_2,\ldots,x_n) = \left\{\underset{x_1}{\operatorname{argmin}} f(x_1,\ldots), \underset{x_2}{\operatorname{argmin}} f(x_2,\ldots),\ldots,\underset{x_m}{\operatorname{argmin}} f(\ldots,x_m)\right\} \tag{4}$$

where $\{x_1, x_2, \ldots, x_m\}$ are disjoint subvectors of decision vector x and $2 \leqslant m \leqslant n$. Function $f(x)$ here can be fully separable when subvectors $\{x_1, x_2, \ldots, x_m\}$ are all one-dimensional, i.e., $m = n$.

Griewank's function is a classical example of partially separable functions, which is defined as [36]:

$$f(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right), \tag{5}$$

where the $f(x)$ has only one global minimum at the point $(0, 0, \ldots, 0)$.

Definition 4 Function $f(x)$ of n decision variables is called *partially additively separable* having m independent subcomponents defined as [33, 35]:

$$f(x) = f_1(x_1) + f_2(x_2) + \ldots + f_m(x_m) \tag{6}$$

where x_i represents mutually exclusive decision vectors of f_i.

An example of a partially additive separable function is 7-non-separable, 1-separable Shifted and Rotated Rastrigin Function, which is defined as:

$$f(z) = \sum_{i=1}^{|s|-1} w_i f(z_i) + f\left(z_{|s|}\right), \tag{7}$$

where $s = \{50, 25, 25, 100, 50, 25, 25, 700\}$;

$$D = \sum_{i=1}^{|s|} s_i = 1000;$$
$$y = x - x^{opt};$$
$$y_i = y\left(\mathcal{P}_{[c_{i-1}+1]} : \mathcal{P}_{[c_i]}\right), i \in \{1, \ldots, |s|\};$$
$$z_i = \Lambda^{10} T_{asy}^{0.2}\left(T_{osz}\left(R_i y_i\right)\right), i \in \{1, \ldots, |s| - 1\};$$
$$z_{|s|} = \Lambda^{10} T_{asy}^{0.2}\left(T_{osz}\left(y_{|s|}\right)\right);$$
$R_i : a|s_i| \times |s_i|$ rotation matrix;
$x \in [-5, 5]^D$; and,
global optimum: $f\left(x^{opt}\right) = 0$.

Definition 5 Function $f(x)$ is a *n-non-separable* function if n of its decision variables x_i are not independent [32].

Function $f(x)$ is *fully-non-separable* if any two of its decision variables x_i are not independent, i.e., every pair of the decision variables interact with each other [33, 35].

The extended Rosenbrock function is an example of a non-separable function, which is defined as [32]:

$$f(x) = \sum_{i=1}^{n-1}\left[\left(1 - x_i^2\right) + 100\left(x_{i+1} - x_i^2\right)\right] \tag{8}$$

where the function has exactly 1 minimum at point $(1, 1, 1)$ for $n = 3$, and 2 minima for $4 \leqslant n \leqslant 7$ when the function gradient equals to zero.

Review on problem decomposition approaches using CC

The critical step of a cooperative co-evolution (CC) technique is how to decompose the problem into subproblems [37]. Generally speaking, a CC algorithm should decompose a large problem into many subproblems with minimal interdependency between them. The first problem decomposition methods using CC used two simple approaches: one-dimensional-based and splitting-in-half strategies [38]. The former type decomposes an n-dimensional problem into n one-dimensional subproblems. This strategy is quite simple and can be effective for separable problems. However, this does not consider the interdependencies between subproblems. Therefore, it is not suitable for handling non-separable problems satisfactorily. In comparison, the splitting-in-half strategy always decomposes an n-dimensional problem into two equal $\frac{n}{2}$-dimensional subproblems. However, when n is large enough, splitting the problem into $\frac{n}{2}$-dimensional subproblems will still be large resulting computational difficulties [39].

Considering the interdependencies between subproblems, a CC framework, EACC-G, was proposed for high-dimensional optimization problems based on random grouping (RG) [39]. This framework randomly divides the decision variables of the high-dimensional problem into a number of groups with predefined group size. It provides a non-zero probability of assigning interacting variables into the same group. In RG, every decision variable has an equal probability of being assigned to any of the subcomponents, and at the beginning of each cycle, the entire process is repeated. This results in a better performance than one-dimensional-based and splitting-in-half strategy decomposition methods for the CC framework. However, the problem here is to define the optimal group size, which is problem-dependent. A small group size may be suitable for separable problems. A larger group size can be useful for non-separable problems, because this can provide a higher probability grouping for more interacting decision variables into the same group. Furthermore, the selection of group size can be difficult at different stages of optimization for a single problem. A small group size can be helpful for finding solution regions faster at the initial stages; however, for later stages, a large group size is preferable, containing more global information for the subproblems. Hence, the appropriate group size depends on the objective function, optimization, and decision variables. To obtain the appropriate group size, a multilevel CC framework (MLCC) was proposed by the same research group in 2010 [40]. In MLCC, a decomposer pool is built based on different group size and using random grouping strategy, where each decomposer in the pool has various interaction levels between decision variables. The evolution process is divided into several cycles, and at the start of each cycle, a decomposer is selected from the pool depending on their record of performance. The selected decomposer is used to decompose the problem into subproblems having decision variables. The record of performance of each decomposer is updated at the end of each cycle. Following this strategy, MLCC is able to self-adapt to an appropriate interaction level regardless of the decision variables, objective function, and optimization stages. To further improve performance, MLCC also includes a weight vector to co-adapt subcomponents. To group interacting variables together, a variable interaction learning (VIL) method was proposed that accounts all decision variables as independent initially and place them into a separate group. It further investigates the variable interactions iteratively, and when interactions are identified by a pairwise interaction between variables

using a non-monotonicity detection, it merges variables into the same group [31]. This method identifies the subcomponents structure and adjusts the component size using two phases: learning and optimization.

Although DECC-G and MLCC provide opportunities to decompose a large problem into subproblems efficiently, they suffer from two major weaknesses, which degrade algorithmic performance. The first one is the decrease of probability to group interacting variables in a subproblem when the number of interacting variables is increased. The second one is that the adaptive weighting in MLCC is not important in the entire process and sometimes fails to improve the quality of the solution and also increases the number of fitness evaluations. To improve the performance of MLCC, a more frequent random grouping (DECC-ML) method has been proposed by Omidvar et al. [14]. The DECC-ML method reduces the number of fitness evaluations to find a solution without deteriorating solution quality. In this method, to choose a decomposer, a uniform random number generator has been used, unlike MLCC, which uses a sophisticated probability formula. After DECC-ML, Omidvar et al. [41] also proposed another decomposition method called *Delta Grouping*. When two interacting variables are grouped into different subcomponents, each variable can have a limit to be improved towards the optimal value in a non-separable function. They calculated the amount of change called *delta value* in each decision variable in every cycle, which can identify interacting variables. Delta values measure the averaged difference for a specific variable over the entire population to identify interacting variables. The assumption here is to have smaller delta values for interacting variables. The decision variables are sorted based on delta values, and variables with smaller delta values are grouped into the same subcomponent. Compared to previous techniques, delta grouping is effective to group interacting variables. However, this grouping method performs poorly if the objective function has more than one non-separable subcomponent.

A metamodel-based decomposition method called *high-dimensional model representation (HDMR)* has also been proposed, which considered two variables non-separable when they have a cooperative effect on the approximated second-order HDMR model [42]. HDMR uses a map of linkage between input and output system variables. For large-scale fully separable continuous functions, Omidvar et al. introduced a decomposition approach, *multilevel softmax (MLSoft)*, in 2014 [43]. MLSoft is basically a modified MLCC algorithm proposed based on reinforcement learning for adaptively finding the proper subcomponent size. The experimental analysis concludes that a subcomponent size should be as small as possible. Because it is not possible to predict the capacity of an optimizer in advance, optimal subcomponent size still requires empirical analysis.

Random grouping and delta grouping decomposition methods decompose an n-dimensional problem into k s-dimensional subproblems. However, the major drawback of these approaches is to determine the value of k or s. When the objective function has large groups of interacting variables, a small s value can decrease algorithmic performance. In contrast, when the objective function has several small groups of interacting variables, a large s value cannot ensure proper utilization of decomposition techniques. Although MLCC [40] solves the problem of specifying the value of s through the use of a decomposer pool with a set of possible s values, it still needs to determine the set of potential s values. In addition, selecting an s value in MLCC divides the decision

variables into a set of equally-sized subcomponents, which is unlikely to have equally-sized interacting groups for most real-world problems. To overcome these limitations, an automatic decomposition method, differential grouping (DG) was introduced by Omidvar et al. in 2013 [44]. DG starts investigating the interaction of the first decision variable with the rest of the variables in a pairwise fashion based on the definition of partially additively separable functions. When DG identifies an interaction between the first variable, and any variable, it removes that variable from the set of decision variables and places it in a subcomponent. This process is iterated until all other interactions between the rest of the variables and the first variable is identified and the first subcomponent is built. When no interaction is found, the variable is assumed separable. The entire process is iterated for the rest of the variables to find interactions for all decision variables. Even though DG achieves promising performance in decomposing a large problem into multiple subproblems by grouping interacting variables together, it has some drawbacks, namely, not detecting overlapping functions, being slow to check all interactions, requiring a threshold parameter, and being sensitive to the choice of the threshold parameter.

DG can identify the decision variables, which interact directly, and to identify the indirect interactions between the decision variables, an extension of DG, *extended DG (XDG)* was proposed by Yuan et al. in 2015 [45]. In XDG, first the variables with direct interactions are grouped to the subcomponents and then the overlappings between subcomponents are detected. When an overlap is detected, the subcomponents with same decision variables are combined. XDG may perform poorer in terms of identifying the interaction among multiple variables. Besides, XDG cannot model the decomposition problem formally, and it inherits the sensitivity issue of DG. To overcome these drawbacks, in 2016, Yingbiao et al. [15] proposed another improvement of DG, called *graph-based DG (gDG)*. This gDG starts by identifying all separable variables and allocating them into the same group. Here, each decision variable is considered as a vertex, and an edge is an interaction between two variables. A graph is then constructed from this arrangement to model the decomposition problem, and the connected components are identified.

In 2016, Mei et al. [46] proposed a global differential grouping (GDG) approach to improve the accuracy of the DG algorithm by adopting the DG mechanism and by maintaining the global information in terms of the interaction matrix between decision variables. In addition, GDG also considers computational error. The authors identified three drawbacks of DG: (1) the comparison between the variables is incomplete; (2) it tends to miss several interactions between decision variables, being the grouping performance sensitive to a threshold parameter; and (3) it cannot work appropriately with fully separable functions or when the functions have a large portion of separable variables. To overcome these limitations, GDG uses three different measures: interaction, independence, and interaction and independence combined. GDG can detect variable dependency and can isolate them into the same groups more accurately. GDG overcomes the sensitivity issue of DG by considering computational errors. However, it is not suitable for imbalanced functions because of the global parameter to identify all interactions.

DG uncovered variable interdependency for making near-optimal decomposition with the cost of ignoring indirect non-separable interactions. XDG and GDG solved the indirect non-separability problem; however, they both need high computational cost

for the interaction identification process. This leads to a suboptimal performance in the following optimization process. The identification process required substantial computational time because of using a pairwise fashion to detect the interactions by most of the algorithms. To address this, a *fast interdependency identification (FII)* algorithm was proposed by Xiao-Min et al. in 2016 [11]. FII performs the decomposition by identifying the interdependency information for separable and non-separable variables. When non-separable variables are discovered, their interdependency is further investigated. FII requires no complete interdependency for non-separable variables, because it avoids interdependency identification in a pairwise fashion. As a result, FII can save a significant number of fitness evaluations for the following optimization process. The near-optimal decomposition can thus be obtained at a small computational cost.

DG is a breakthrough for large-scale problem decomposition. However, it has a number of drawbacks, and to address these, several improvements have been proposed based on DG. Similarly, another improvement of DG, called *DG2*, has been proposed by the same research group who introduced DG [47]. For fully separable functions, DG2 reduces the number of fitness evaluations by half. This, in turn, supports the algorithm to check every pair of interacting variables at a significantly lower computational cost compared to the previous approaches. The overlapping functions can be effectively identified by testing all pairs of variables interaction. If the DG mechanism is used to develop DG2, a systematic generation of sample points maximize the point reuse, which provides lower bound to detect the interactions. Beyond efficiency, grouping accuracy of DG is also improved significantly with DG2. Since DG2 considers the computational rounding-errors to estimate an appropriate threshold level, it can determine the sensitivity to weak variable interactions. The automatic calculation of the threshold value is useful to deal with imbalanced functions. Furthermore, unlike DG, DG2 does not require an external parameter.

Because previous decomposition algorithms required a large number of fitness evaluations (otherwise failed to obtain proper decomposition results), a historical interdependency based differential grouping (*HIDG*) was proposed in 2018 [12]. Similar to FII, HIDG also uses decision vectors to investigate the interdependencies among vectors in a systematic way. Based on the historical interdependency information, HIDG proposes a criterion to infer interactions for decision vectors without using extra fitness evaluations. To reduce the computational expense of the decomposition algorithm, a recursive differential grouping (RDG) method was proposed by Yuan et al. in 2017 [10]. RDG considers the interaction between decision variables depending on non-linearity identification. It recursively analyzes the interaction between variable x_i and the rest of the variables. When an interaction is detected, the rest of the variables are divided into two groups of equal size. Then the interaction between variable x_i and each group is examined. The process is repeated until all independent variables that interact with x_i are detected and placed into the corresponding subcomponent as x_i. The analytical methods validate the effectiveness of RDG without the need to examine the variable interactions in a pairwise fashion.

Another improvement of DG was proposed to overcome its drawbacks regarding the threshold parameter and identify indirect interaction, called ε-*based DG* (ε-DG) [8]. In general, ε is a predefined threshold value, and ε is zero [44]. According to DG,

when $delta > \varepsilon$, the decision variables are considered non-separable; otherwise, they are separable. However, authors of ε-DG experimented and observed that $\varepsilon = 0$ is ineffective because $\varepsilon > 0$ also occurs with separable variables, i.e., setting $\varepsilon = 0$ can classify a few decision variables to be non-separable, which are supposed to be separable. ε-DG detects that $delta > 0$ for two separable variables are resulted from a computational error. Hence, ε-DG first checks whether $delta > 0$ is resulted from an interaction or computational error. If it results from a computational error, *delta* is set to zero, which can solve the configuration issue of ε. In ε-DG, an $n \times n$ DG matrix (DGM) is constructed to represent the interactions between every pair of decision variables. The DGM is able to identify both direct and indirect interactions by setting an element of DGM to zero when there is no interaction between two associated variables; otherwise, it sets to a non-zero value.

Based on the concept of partial derivatives of multivariate functions, GDG has been proven to be an effective grouping strategy in automatically resolving the problem by the global information between variables. However, the grouping results of GDG are no longer updated once GDG decomposed a problem into subproblems and it is not automatically adjusted throughout the algorithmic evolution process. Hence, to resolve the problem by updating the grouping results throughout the evolutionary process, an improvement of GDG was proposed by Wu et al. in 2018 [48], called *Dynamic GDG (D-GDG)*. Based on GDG, in D-GDG, the original data matrix of interacting variables is obtained from the general idea of the partial derivative of multivariate functions. The fuzzy clustering technique is used to discern the dynamic clustering of decision variables during the standardization of the original data matrix and to build the fuzzy relation matrix. The size of each group of interacting variables is limited by the lower and upper bounds of the variable grouping scale. The grouping of variables has been adaptively adjusted according to the algorithmic state throughout the optimization process.

In terms of time complexity, RDG is a very effective decomposition method. However, to identify whether two subsets of variables interact, a proper parameter setting is required to determine a threshold value. Furthermore, one global threshold value may not be sufficient for determining interactions of variables in subcomponents having a dissimilar contribution to the fitness value. To solve these problems, a decomposition method inspired by DG2 and RDG was proposed, called *RDG2* [49]. In RDG2, the threshold value is adaptively estimated based on the computational round-off errors of RDG. An upper bound of the round-off errors can be adequate to distinguish between separable and non-separable variables of large-scale benchmark problems. Besides, experimental results have shown a higher decomposition accuracy of RDG2 compared to RDG and DG2.

All the preceding decomposition algorithms usually group linked or interacting variables into the same group. However, this does not tend to reduce the size of the original problem. Moreover, overlapping problems pose a further challenge to the decomposition algorithms because of the dependency of the components on each other. To address these issues, the RDG decomposition algorithm has been further modified for decomposing overlapping problems. This new algorithm, *RDG3*, breaks the linkage at variables, which is shared by more than one component. The

performance of RDG3 has been evaluated by overlapping benchmark problems, which confirmed its superiority over RDG and other decomposition algorithms [50]. Table 2 presents a summary of the papers reviewed on problem decomposition techniques with key features.

From the literature, it has been observed that problem decomposition techniques fall into two categories [11, 51]:

Static decomposition A problem is decomposed into subproblems before the evolutionary process of CC starts, and the decomposed subproblems are fixed for the rest of the generations [11]. In the original CC implementation (CCGA), a problem of n-dimension was divided into n one-dimensional subproblems [34]. With this one-dimensional-based decomposition, CCGA performed better on separable problems than on non-separable problems (in CCGA, Rosenbrock was used as a non-separable problem). This concluded two major problems of static one-dimensional-based decomposition. First, the search capacity of CC may degrade drastically if there are interdependencies exist between the subcomponents. Second, one-dimensional decomposition may waste computational resources during the evolutionary process. The splitting-in-half static decomposition method cannot handle non-separable problems properly either [38]. By exploring the interdependency information in the problem, a number of detection-based static decomposition techniques proposed to make the near-optimal decomposition. Examples of such static decomposition methods include CCVIL [31], DG [44], XDG [45], GDG [46], RDG [10], D-GDG [48], RDG2 [49], and RDG3 [50]. These methods are also called automatic decomposition strategies as they undertake the variable interactions to group the variables [10].

Dynamic decomposition A problem is decomposed into multiple subproblems at the start. However, it has the ability to adapt the subcomponents during the evolutionary process, i.e., the decision variables can be regrouped into subcomponents in each cycle of the evolutionary process [11]. RG is a typical example of a dynamic decomposition technique [39]. In RG, at the start of each evolutionary process, two interacting decision variables have a non-zero probability of being grouped in the same subcomponent. However, as the number of interacting variables increases, RG becomes incapable of grouping all the interacting variables in the same group. Other approaches to improve the performance of dynamic decomposition techniques are also available including MLCC [40], DECC-ML [14], and delta grouping (DECC-D) [28].

The previous extensive literature review on problem decomposition approaches using CC found that the problem decomposition techniques have been widely used, improved, and evaluated on a range of benchmark functions. Apart from these strategies, a number of decomposition methods have also been studied in the literature with CC for some real-world problems. For example, a random-based dynamic grouping method was proposed to group variables for multi-objective optimization problems, which uses a decomposer pool similar to MLCC [40]. In this method, the decomposer pool consists of several groups of varying sizes, and the size is determined by C-metric based on historical performance [52]. A weighted random (WR) grouping strategy proposed for large-scale crossing waypoints location problem (a fully non-separable and non-differentiable problem) in air route networks where the space of the problem was updated by extreme weather conditions, breakdowns, and delayed aeroplanes [53]. A Markov

Table 2 Summary of papers reviewed on problem decomposition techniques

Algorithm	Methods used	Key features	Limitations
One-dimensional-based [34]	An n-dimensional problem into n one-dimensional subproblems	Quite simple and effective for separable problems	Does not consider the interdependencies between subproblems; unlikely to handle the non-separable problems satisfactorily
Splitting-in-half strategy [38]	An n-dimensional problem into two equal $\frac{n}{2}$-dimensional subproblems	Simple and effective for separable problems like one-dimensional-based approach	When n is large enough, splitting the problem into $\frac{n}{2}$-dimensional subproblems will still be large; computationally expensive; handles even number of dimensions only
DECC-G [39]	Randomly divides the decision variables in the high-dimensional problem into a number of groups with predefined group size	Provides a non-zero probability of assigning interacting variables into the same group	Defining optimal group size, which is problem-dependent; selection of group size; a decrease of probability to group interacting variables in one subproblem when the number of interacting variables is increased
MLCC [40]	A decomposer pool of variable group size and random grouping	Self-adaptation to appropriate interaction level despite decision variables, objective function, and optimization stages	Adaptive weighting in MLCC is not important in the entire process and sometimes fails to improve the quality of solution and also causes an extra number of fitness evaluations
CCVIL [31]	Incremental group size; two phases: learning and optimization	Variable interactions detection in one stage and optimization of groups in another stage in the same framework	Heavy computations due to pairwise interaction check
DECC-ML [14]	More frequent random grouping; uniform selection of subcomponent size	Reduces the number of fitness evaluations to find a solution without deteriorating solution quality	Ineffective when the number of interacting variables increases to five or more; probability to group more than two interacting variables into the same subcomponent tends to be zero despite a co-evolutionary cycle
DECC-D [28]	Delta value	Delta value computes the amount of change in each decision variable in every cycle to identify interacting variables	Suffers from low-performance issue when the objective function has more than one non-separable subcomponent
DM-HDMR [42]	Meta-modelling decomposition	Applies the first order RBF-HDMR model for extracting interactions between decision variables	Computationally expensive; validation required on real-world problems
MLSoft [43]	Value function; softmax selection	Modification of MLCC and using reinforcement learning	Not suitable for problems of a non-stationary nature
DECC-DG [44]	Automatic decomposition strategy	Groups interacting variables before optimization and fixes grouping during optimization	Not detecting overlapping functions; slow to check all interactions; requires a threshold parameter; sensitive to the choice of the threshold parameter

Table 2 (continued)

Algorithm	Methods used	Key features	Limitations
XDG [45]	Identifies direct first and then checks indirect interactions separately	The subcomponents with the same decision variables are combined when an overlap between subcomponents is identified; identifies indirect interactions between decision variables	Not modeling the decomposition problem in a formal way; inherits the sensitivity issue of DG; high computational cost for the interaction identification process
DECC-gDG [15]	Decision variable as vertex, interaction between variables as edge, a graph is constructed to model the problem	Identifies all separable problems and allocates them into the same group	Grouping accuracy depends on a variable threshold parameter
GDG [46]	Adopts DG; maintains global information in terms of interaction, interdependence, and interaction and interdependence	Detect variable dependency and can isolate them into the same groups more accurately	Not suitable for imbalanced functions because of a global parameter to identify all interactions; high computational cost for the interaction identification process; grouping results of GDG are no longer updated once GDG decomposed a problem into subproblems
FII [11]	Identifies the interdependency information for separable and non-separable variables; for non-separable further investigation	Requires no complete interdependency information for non-separable variables because it avoids the interdependency identification in a pairwise fashion	The number of fitness evaluations is n^2 for decomposing overlapping problems; identification accuracy is slightly lower than XDG and GDG; performance limitation on imbalance problems
DG2 [47]	A systematic generation of sample points to maximize the point reuse; computational rounding-errors to estimate an appropriate threshold level	For fully separable functions, DG2 reduces the number of fitness evaluations by half; the automatic calculation of the threshold value is useful to deal with imbalanced functions	Neglects the topology information of decision variables of large-scale problems
HIDG [12]	Decision vectors to investigate the interdependencies between vectors	Infers interactions for decision vectors without using extra fitness evaluations	A complete integration of HIDG with a CC framework is yet to be explored
RDG [10]	Interaction between decision variables based on the non-linearity identification	Fitness evaluation takes only $\mathcal{O}(n \log(n))$ time	To identify whether two subsets of variables interact, a proper parameter setting is required to determine a threshold value
(ε-DG) [8]	Delta check for computational errors; DG matrix construction	Identify both direct and indirect interactions by setting an element of DGM to zero or non-zero	Effectiveness of the algorithm has not been evaluated on real-world problems
D-GDG [48]	Data matrix construction from the general idea of the partial derivative of multivariate functions; fuzzy clustering technique	The grouping of variables has been adaptively adjusted according to the algorithmic state throughout the optimization process	Effectiveness of the algorithm on large-scale black-box real-world optimization problems needs to be verified
RDG2 [49]	Adaptively estimation of threshold value	Round-off errors are adequate to distinguish between separable and non-separable variables of large-scale benchmark problems	Neglects the topology information of decision variables of large-scale problems
RDG3 [50]	Modified RDG; breaking the linkage at variables	Decomposing overlapping problems	Does not consider weak linkage breaking

random field-based decomposition method proposed to solve the stochastic flexible job shop scheduling problem for minimizing the expectation and variance of makespan [54].

Because the number of existing work on FS using CC is limited, problem decomposition techniques using CC for FS are also limited in the literature. A quantum-inspired self-adaptive CC incorporated into shuffled frog leaping algorithm was proposed by Ding and Wang in 2013 to reduce the number of attributes [55]. In this framework, the attribute sets were divided into subsets using a self-adaptive technique that selects a decomposer from the decomposer pool based on their historical performance records. The performance of this approach was evaluated on global optimization functions and UCI datasets. Recently, in 2019, Wang et al. [29] proposed a two-stage decomposition approach (CCFS/TD) based on CC for FS on high-dimensional classification problem. In the first stage, CCFS/TD decomposes the evolutionary cycle into m levels and performs optimization. In the second stage, the evolutionary cycle of each level is further decomposed into a number of independent processes using a majority voting technique. The effectiveness of the proposed CCFS/TD method has been validated by experiments on 10 benchmark datasets. A study by Sun et al. in 2015 [9] on the selection of decomposition methods for large-scale fully non-separable problems performed experiments on decomposition techniques using RG, delta grouping, DG, and XDG. This study observed that RG is the most effective decomposition method for fully non-separable problems. Because FS problems are fully non-separable, based on the motivation from the study of [9], in the next section of this paper, a random-based decomposition method, called *random feature grouping (RFG)* is proposed with three variants.

A novel feature selection approach

In this paper, a CC-based FS framework is introduced with a proposed random feature grouping (RFG) decomposition technique, Cooperative Co-Evolutionary-Based Feature Selection with Random Feature Grouping (CCFSRFG). The motivation to propose RFG comes from the success of the random grouping decomposition strategy for non-separable function optimization problems and because the feature selection problem is a non-separable optimization problem [9, 13, 14, 39, 56]. CCFSRFG has been applied to the FS problem in Big Data using six widely used ML classifiers, naïve Bayes (NB) [57], support vector machine (SVM) [58], k-Nearest Neighbor (k-NN) [59], J48 [60], random forest (RF) [61], and logistic regression (LR) [62], on seven different datasets from the UCI ML repository.[3] At first, the ML classifiers have been applied to all datasets without reducing the number of features. After that, CCFSRFG with two variants, is applied to all classifiers for reducing the number of features of the datasets. A penalty-based wrapper objective function defined in a previous study has been used as the fitness function in the CCFSRFG framework to deal with the twofold objectives of FS (reducing the dimension of features, and improving classification performance) in process in Big Data. The comparative results have been analyzed based on different performance measures, including accuracy, sensitivity, and specificity [63].

[3] http://archive.ics.uci.edu/ml/.

Cooperative co-evolution

In 1994, Potter and De Jong [34] introduced the cooperative co-evolution (CC) technique to solve optimization problems, which are large-scale and complex. This technique uses a divide-and-conquer approach to divide a large problem into multiple subproblems, and iteratively evolves the interacting co-adapted subproblems to build a complete solution.

The CC technique consists of decomposing an n-dimensional problem of search information $S = \{1, 2, \ldots, n\}$ into m subproblems $\{S_1, S_2, \ldots, S_m\}$ [34]. Each subproblem of the n dimensions represents a new search space $SP^{(i)}$ for a particular problem, where the rest of the dimensions n_j, with $j \neq S_i$, are kept fixed. Hence, the entire search space is decomposed into m subproblems with lower dimensions and that can be handled using any population-based evolutionary computational approach. These subproblems can be optimized individually, and communication between them is required only to evaluate the objective (fitness) function f. This implies that a candidate solution in search space $SP^{(i)}$ contains a few elements (comprising an individual I) of the n-dimensional problem ($I \in SP$). Therefore, in CC, a common n-dimensional context vector v is required to build using a collaborative individual (e.g., the current best individual) from each subproblem. A candidate solution to the problem is then formed by a cooperative algorithm to evaluate an individual in a subproblem, a candidate solution is built by joining representative collaborators from the context vector. Potter and De Jong decomposed an n-dimensional problem into n 1-dimensional subproblems in their original CC idea. In general, the n-dimensional problem can be decomposed into m subproblems with same dimension, i.e., $n_m = n/m$ [64]. Following this, a CC technique can be formally defined as follows [65]:

An n-dimensional problem is decomposed as:

$$S_i = \{(i-1) \times n_m + 1, \ldots, i \times n_m\},$$

and the context vector is built as:

$$v = \underbrace{(v_1^{(1)}, \ldots, v_{n_m}^{(1)}}_{v^{(1)}}, \underbrace{(v_1^{(2)}, \ldots, v_{n_m}^{(2)}}_{v^{(2)}}, \ldots, \underbrace{(v_1^{(m)}, \ldots, v_{n_m}^{(m)})}_{v^{(m)}}{}^T,$$

where $v^{(i)}$ is the n_m-dimensional problem, which represents the collaborative individual from the i-th subproblem (e.g., the current best individual in $SP^{(i)}$):

$$v^{(i)} = \left(v_1^{(1)}, v_1^{(2)}, \ldots, v_{n_m}^{(1)}\right)^T.$$

Given the j-th individual $I^{(i,j)} \in SP^{(i)}$ of the i-th subproblem:

$$I^{(i,j)} = \left(I_1^{(i,j)}, I_2^{(i,j)}, \ldots, I_{n_m}^{(i,j)}\right)^T.$$

The fitness function of this problem is given by $f\left(v^{(i,j)}\right)$, where $v^{(i,j)}$ is defined as:

$$v^{(i,j)} = \underbrace{(v_1^{(1)}, \ldots, v_{n_m}^{(1)}}_{v^{(1)}}, \underbrace{(I_1^{(2)}, \ldots, I_{n_m}^{(i,j)}}_{I^{(i,j)}}, \ldots, \underbrace{(v_1^{(m)}, \ldots, v_{n_m}^{(m)})}_{v^{(m)}}{}^T.$$

In general, the fitness of $v^{(i,j)}$ is evaluated on context vector v by replacing the elements of the individual from the i-th subproblem having the representative elements of individual $I^{(i,j)}$.

CC is therefore comprised of three main phases: (1) problem decomposition, (2) subproblem evolution, and (3) collaboration and evaluation.

Problem decomposition

The first phase of CC is how to decompose a large problem into subproblems generally depends on the problem structure [30]. If the problem is decomposed statically, it will have one element in each subproblem, whereas if the problem is decomposed dynamically, the grouping of elements into subproblems will be different than with the former. One objective of this paper is to review existing CC-based problem decomposition techniques and propose a suitable decomposition method for feature selection. Hence, CC problem decomposition is described in detail in "Literature review" section.

Subproblem evolution

Once the problem is decomposed into subproblems, each subproblem is allocated to a subpopulation. Each subpopulation is optimized individually either through a homogeneous or a heterogeneous evolutionary optimizer [30]. Optimizations of subpopulations can be carried out either sequentially or in parallel. In the sequential case, only one subpopulation evolves in each generation [66]. In contrast, in the parallel case, all subpopulations evolve in each generation simultaneously [67]. The most widely used evolutionary optimizer in this area is a genetic algorithm (GA), whereas differential evolution (DE) [68] is the most effective optimizer for CC.

Collaboration and evaluation

After subproblem optimization, subpopulations cooperate to build a complete solution to the problem. The fitness (objective function) of an individual is evaluated by selecting a collaborator individual from each subpopulation. The performance of this collaboration is specified as the fitness value to a individual being evaluated. Individuals having the best collaborating performance are joined together to discover the final solution to the problem at the end of a CC process [30]. $1+1$ collaboration [37], the $1+N$ collaboration model [69], and reference sharing (RS) [30] are examples of collaboration models used in CC.

Methodology

In this paper, a CC-based feature selection framework in Big Data with a random feature grouping (RFG) is introduced, called CCFSRFG. Considering correlations to be linearly associated with a dataset's records, for example, in a network traffic dataset, a correlation measure, such as a linear correlation coefficient, can be applied to measure the linear dependency between two random features. However, there are many real-world problems where the correlation between the features can be nonlinear as well. Therefore, a linear correlation study cannot measure the nonlinear dependency between two features. As a consequence, irrespective of whether the dependency between two features is linear or nonlinear, selecting a subset of features from the dataset that maximize

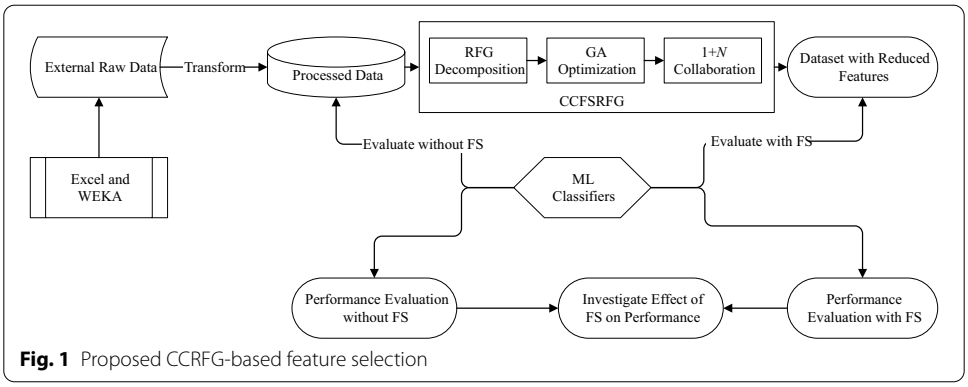

Fig. 1 Proposed CCRFG-based feature selection

classification accuracy is more appropriate [70]. Accordingly, the feature selection framework, CCFSRFG, with FRG as a decomposer, is proposed to select a suitable subset of features without considering its correlation. The RFG will increase the chance of optimizing interacting features together. The proposed methodology of CCFSRFG is displayed in Fig. 1.

The main idea behind this new RFG-based cooperative co-evolution feature selection (CCFS) framework is to dynamically decompose the feature vector into ms-dimensional subdatasets ($n = m \cdot s$). The fundamental difference between this new approach and the previously studied CCEAFS [19] approach is that CCFSRFG decomposes the feature vector dynamically to increase the probability of grouping interacting features together for subsequent optimization phase, which can lead to improving solution quality; while CCEAFS decomposes the feature vector statically starting from the feature indexed at 1, which cannot ensure the grouping of interacting features and may result in a low-quality solution. Formally, the proposed CCFSRFG can be described as follows:

A dataset D consisting of n features, i.e., $D = \{f_1, f_2, f_3, \ldots, f_n\}$. D is decomposed randomly into m subdatasets with $s (s < n)$ features in each subdataset:

$$
\begin{aligned}
D_1 &= \{f_{i_1}, f_{i_2}, \ldots, f_{i_s}\} \\
D_2 &= \{f_{i_1}, f_{i_2}, \ldots, f_{i_s}\} \\
&\ldots \\
D_m &= \{f_{i_1}, f_{i_2}, \ldots, f_{i_s}\}.
\end{aligned}
$$

Each subdataset is represented using a subpopulation in CCFSRFG. Here, s is the number of features in each individual (i.e., s features of a subdataset). Consider the size of each subpopulation (sp) is sz. An example of subpopulation sp_1 consisting s individual can be the following:

$$
\begin{aligned}
ind_1 &= \{0, 1, 1, 0, \ldots, 1\} \\
ind_2 &= \{1, 1, 1, 0, \ldots, 0\} \\
&\ldots \\
ind_{sz} &= \{0, 1, 1, 1, \ldots, 1\}.
\end{aligned}
$$

1 in an individual indicates that the feature in the corresponding is selected for the feature subset selection; 0 indicates that the feature is not selected for the feature subset selection. An individual in any subpopulation is evaluated by joining collaborators (i.e.,

individuals) from other subpopulations. For example, to evaluate individual ind_1 in subpopulation sp_1, s.t. collaborator ind_3 from subpopulation sp_2 and collaborator ind_2 from subpopulation sp_3, these three individuals are joined together to build a complete solution for the dataset with a reduced number features. Consider a random decomposition of 9 features into three subpopulations ($s = 4$), is assumed with $sp_1\{ind_1\}=\{f_3,f_9,f_7,f_2\}$, $sp_2\{ind_2\}=\{f_6,f_1,f_5,f_8\}$, and $sp_3\{ind_4\}=\{f_4\}$. If features $\{f_7,f_2\}$ from $sp_1\{ind_1\}$, $\{f_1,f_5\}$ from $sp_2\{ind_2\}$, and $\{f_4\}$ from $sp_3\{ind_4\}$ are selected because of a binary (0 or 1) representation of features, the complete solution with sorted feature indices can be defined as follows:

$$solution = \{f_1,f_2,f_4,f_5,f_7\}.$$

The solution with this reduced number of features is then evaluated by the ML classifiers to measure accuracy, sensitivity, and specificity performance. The best individual with a reduced number of features and the highest classification accuracy is achieved by a penalty-based wrapper objective function introduced in the previously studied CCEAFS approach [19]. The objective function for this is defined as:

$$f = w_1 * f_1 - w_2 * f_2, \tag{9}$$

where $f_1 = T_c/T$ and $f_2 = S/N$.

f is the overall objective function; w_1 and w_2 are two control parameters for the objective functions f_1 and f_2, which are used to adjust the penalty term for f_1 and f_2, with $w_1 + w_2 = 1$; T is the total number of test or train samples in the dataset (the test or train samples depend on the classification mode of using cross-validation or the supplied test set);

T_c is the number of correctly classified instances in the test or train samples; S is the number of features selected in the subset; and N is the total number of features in the dataset.

For CCFSRFG-1, the best individuals from other subpopulations are used as collaborators from generation 1 onwards, whereas in the case of CCFSRFG-2, for all generations, random individuals from other subpopulations are used as collaborators. The process continues until it reaches a fixed number of generations, or until no better fitness is achieved over the generations.

Datasets from the UCI ML repository have been gathered, and Microsoft Excel and *WEKA*[4] used to preprocess the datasets. The preprocessing stage mainly deals with the dataset conversion from CSV to the Attribute-Relation File Format (ARFF) to make it compatible with the experimental environment. The datasets were evaluated using six ML classifiers: NB, SVM, *k*-NN, J48, RF, and LR. The performance of these classifiers was measured in terms of accuracy, sensitivity, and specificity. CCFSRFG was employed to the datasets to decrease the number of features. The datasets with reduced dimensions were then evaluated using the same six ML classifiers, and the performance of each classifier was measured using the aforementioned metrics. The performance results obtained by the classifiers with and without FS were investigated, and the effect of FS on the performance of the classifiers was analyzed.

[4] https://www.cs.waikato.ac.nz/ml/weka/.

Algorithms 1–2 are the pseudocodes of the proposed CCFSRFG framework with two implementations of RFG. The fundamental difference between algorithms CCFSRFG-1 and CCFSRFG-2 has been indicated by line 32 in Algorithm 1 and line 26 and 33 in Algorithm 2, respectively. A JAVA-based implementation of the framework is available at GitHub.[5]

In the next section, a new novel random feature grouping (RFG) is proposed with its three variants to used as the decomposer for CCFSRFG for decomposing the dimension of a dataset into lower-dimensional subdatasets.

A novel random feature grouping technique

The n 1-dimensional decomposition strategy groups a single feature into a single subcomponent. For example, if there are 9 features in a dataset, the 1-dimensional grouping will be like the following for i individuals for every generation:

$$
\begin{aligned}
&ind_0 : \{f_1\}, \{f_2\}, \{f_3\}, \{f_4\}, \{f_5\}, \{f_6\}, \{f_7\}, \{f_8\}, \{f_9\}\\
&ind_1 : \{f_1\}, \{f_2\}, \{f_3\}, \{f_4\}, \{f_5\}, \{f_6\}, \{f_7\}, \{f_8\}, \{f_9\}\\
&\ldots\\
&ind_i : \{f_1\}, \{f_2\}, \{f_3\}, \{f_4\}, \{f_5\}, \{f_6\}, \{f_7\}, \{f_8\}, \{f_9\}.
\end{aligned}
$$

Algorithm 1 CCFSRFG-1

```
1: Read the dataset to get the number of features f;
2: Initialize nSubPop, subPopSize, gen;
3: Calculate the length of individual: l = f/nSubPop;
4: Dynamically (using Algorithm 3 RFG-1) decompose f features into nSubPop having l features each;
5: gen = 1;
6: for x = 1 to nSubPop do
7:     for y = 1 to subPopSize do
8:         Initialize individual with 0 and 1;
9:     end for
10: end for
11: for x = 1 to nSubPop do
12:     for y = 1 to subPopSize do
13:         Find random collaborators for each individual;
14:     end for
15: end for
16: for x = 1 to nSubPop do
17:     for y = 1 to subPopSize do
18:         Evaluate all individuals according to (9) and sort them in descending order of fitness;
19:     end for
20: end for
21: for x = 1 to nSubPop do
22:     Find the best individual from each subpopulation and sort them in descending order of fitness;
23: end for
24: Select the globally best solution and store the optimal feature subset with the highest fitness value;
25: while gen ⩽ gen_max do
26:     gen = gen + 1;
27:     for x = 1 to nSubPop do
28:         Evolve each subpopulation using a genetic algorithm;
29:     end for
30:     for x = 1 to nSubPop do
31:         for y = 1 to subPopSize do
32:             Find the best individuals from the previous generation as collaborators for each individual in the current
                generation;
33:         end for
34:     end for
35:     for x = 1 to nSubPop do
36:         for y = 1 to subPopSize do
37:             Evaluate all individuals according to (9) and sort them in descending order of fitness;
38:         end for
39:     end for
40:     for x = 1 to nSubPop do
41:         Find the best individual from each subpopulation and sort them in descending order of fitness;
42:     end for
43:     Select the globally best solution and store the optimal feature subset with the highest fitness value;
44: end while
```

[5] https://github.com/bazlurrashid/cooperative-coevolution/tree/CCFSRFG/.

For each generation of an evolutionary process, the grouping will be fixed. Similarly, an $m \cdot s$-dimensional dataset ($m = 3$ subcomponents, $s = 4$ features in a subcomponent) can have the following grouping with 9 features for all individuals in every generation:

$$ind_0 : \{f_1, f_2, f_3, f_4\}, \{f_5, f_6, f_7, f_8\}, \{f_9\}$$
$$ind_1 : \{f_1, f_2, f_3, f_4\}, \{f_5, f_6, f_7, f_8\}, \{f_9\}$$
$$\ldots$$
$$ind_i : \{f_1, f_2, f_3, f_4\}, \{f_5, f_6, f_7, f_8\}, \{f_9\}.$$

Similar to n 1-dimensional grouping, a $m \cdot s$-dimensional grouping also maintains the same group components throughout the evolutionary process.

The proposed random feature grouping (RFG) can group features in dataset into different groups using three different ways based on the motivation from [9, 14, 35, 39].

Algorithm 2 CCFSRFG-2

```
Read the dataset to get the number of features f;
Initialize nSubPop, subPopSize, gen;
Calculate the length of individual: l = f/nSubPop;
Dynamically (using Algorithm 3 RFG-1) decompose f features into nSubPop having l features each;
gen = 1;
for x = 1 to nSubPop do
    for y = 1 to subPopSize do
        Initialize individual with 0 and 1;
    end for
end for
for x = 1 to nSubPop do
    for y = 1 to subPopSize do
        Find random collaborators for each individual;
    end for
end for
for x = 1 to nSubPop do
    for y = 1 to subPopSize do
        Evaluate all individuals according to (9) and sort them in descending order of fitness;
    end for
end for
for x = 1 to nSubPop do
    Find the best individual from each subpopulation and sort them in descending order of fitness;
end for
Select the globally best solution and store the optimal feature subset with the highest fitness value;
while gen ⩽ gen_max do
    Dynamically (using Algorithm 3 RFG-2) decompose f features into nSubPop having l features each;
    gen = gen + 1;
    for x = 1 to nSubPop do
        Evolve each subpopulation using a genetic algorithm;
    end for
    for x = 1 to nSubPop do
        for y = 1 to subPopSize do
            Find random collaborators for each individual except the first individual from each subpopulation which
            is being used as elitism and has different features in the individual from the previous generation;
        end for
    end for
    for x = 1 to nSubPop do
        for y = 1 to subPopSize do
            Evaluate all individuals according to (9) and sort them in descending order of fitness;
        end for
    end for
    for x = 1 to nSubPop do
        Find the best individual from each subpopulation and sort them in descending order of fitness;
    end for
    Select the globally best solution and store the optimal feature subset with the highest fitness value;
end while
```

RFG-1

An n-dimensional dataset is decomposed randomly into m-subcomponents and the features in the subcomponents are fixed for every generation. For example, if a dataset has 9

features, the RFG-1 groups 9 features into 3 subcomponents ($s = 4$) features) in the first generation for i individuals the following way:

$$ind_0 : \{f_2,f_3,f_6,f_5\}, \{f_7,f_1,f_4,f_9\}, \{f_8\}$$
$$ind_1 : \{f_2,f_3,f_6,f_5\}, \{f_7,f_1,f_4,f_9\}, \{f_8\}$$
$$\dots$$
$$ind_i : \{f_2,f_3,f_6,f_5\}, \{f_7,f_1,f_4,f_9\}, \{f_8\}.$$

This will be maintained within the same group components for all individuals and for all generations (from 0 to g), i.e., the groups will be fixed for further generations and only a single time at the start of an evolutionary process, the problem will be decomposed randomly into subproblems. Theorem 1 states the probability of grouping interacting features using RFG-1.

Theorem 1 *Given I individuals, the probability of grouping v interacting features $\{f_1,f_2,\dots,f_v\}$ into a single subcomponent for at least i individuals in any generation based on the probability function of more frequent grouping for function optimization problems can be defined as* [14]:

$$P(F \geqslant i) = \sum_{r=i}^{I} \binom{I}{r} \left(\frac{1}{s^{v-1}}\right)^r \left(1 - \frac{1}{s^{v-1}}\right)^{I-r}, \tag{10}$$

where I is the number of individuals, v is the number of interacting features, s is the number of subcomponents (i.e., number of subproblems or number of subpopulations), random variable F is the number of times v interacting features are grouped into the same subcomponent, $F \geqslant i$, and $i \leqslant I$.

Here, the term $\left(\frac{1}{s^{v-1}}\right)^r$ indicates the probability of grouping v features into a single subcomponent for the first r individuals, and the term $\left(1 - \frac{1}{s^{v-1}}\right)^{I-r}$ indicates the probability of grouping the same number of v features into a subcomponent for the remaining $I - r$ individuals. The proposed RFG-1 approach implies that the grouping components will remain the same for every individual, and this is why the value of r is 1 here.

Therefore, the probability of grouping v interacting features $\{f_1,f_2,\dots,f_v\}$ into a subcomponent irrespective of the number of individuals in any generation by RFG-1 is defined as:

$$P(F \geqslant i) = \frac{1}{s^{v-1}}. \tag{11}$$

For example, given $s = 3$ and $v = 4$: $P(F \geqslant i) = \frac{1}{3^{4-1}} = 0.03704$; for $s = 3$ and $v = 5$: $P(F \geqslant i) = \frac{1}{3^{5-1}} = 0.01235$. This means that if the number of interacting features increase, the probability decreases. Figure 2 shows the probability of grouping interacting features randomly into a single subcomponent with the increase in the number of interacting features for any individual in any generation of an evolutionary process.

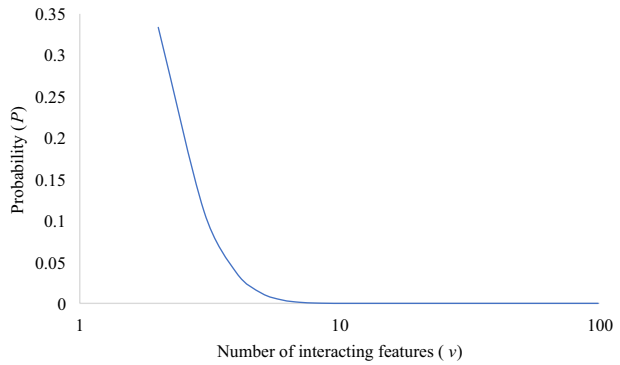

Fig. 2 Given $s = 3$, the probability of grouping interacting features is decreasing with respect to the increasing number of interacting features v

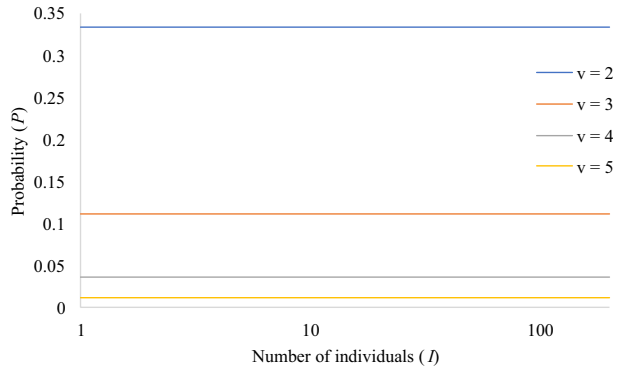

Fig. 3 Given $s = 3$, the probability of grouping interacting features is stable with respect to the increasing number of individuals I for $v = 2, 3, 4, 5$ interacting features irrespective of the number of individuals

It can be observed from the graph that since in RFG-1, the features are grouped randomly only once at the beginning of an evolutionary process, the probability of grouping 9 features tends to zero irrespective of the number of individuals. However, if the number of interacting features is small, the probability of grouping is large, which is independent of the number of individuals shown in Fig. 2. The figure shows that when the number of interacting features is about 10, the probability of grouping these features into a single subcomponent is nearly zero. The probability of grouping interacting features randomly into a single subcomponent in term of the number of individuals in any generation of an evolutionary process is shown in Fig. 3. It can be seen that since the features are grouped randomly only once at the beginning of the evolutionary process, for any number of interacting variables, the probability will not be changed. For example, whether the number of individuals is 0 or 100, for $v = 3$, the probability remains the same.

Algorithm 3 is the pseudocode of the proposed random feature grouping technique

Algorithm 3 RFG-1

Require: f, l;
1: Initialize l;
2: List the feature indices in a list;
3: Apply a shuffling technique to randomize the feature indices;
4: Decompose the shuffled list of feature indices $sublists$ based on the number of features (l) in a subcomponent;
5: Store the value of the number of subcomponent s (i.e., subpopulation) and the size of each subcomponent $genes$;
6: **return** $s, genes, sublists$

(RFG-1).

RFG-2

An n-dimensional dataset is decomposed randomly into m subcomponents in every generation unlike RFG-1, in case of which the dataset is decomposed randomly only once. However, similar to RFG-1, in RFG-2, all individuals will have the same features in the same subcomponents for any given generation. For example, for a dataset with 9 features, if the first generation follows the grouping of features for all individuals as illustrated in the case of RFG-1, the RFG-2 groups 9 groups into 3 subcomponents (s = 4 features) in the second generation randomly as follows:

$$
\begin{aligned}
&ind_0 : \{f_3,f_9,f_7,f_2\}, \{f_6,f_1,f_5,f_8\}, \{f_4\}\\
&ind_1 : \{f_3,f_9,f_7,f_2\}, \{f_6,f_1,f_5,f_8\}, \{f_4\}\\
&\ldots\\
&ind_i : \{f_3,f_9,f_7,f_2\}, \{f_6,f_1,f_5,f_8\}, \{f_4\}.
\end{aligned}
$$

In this strategy, in each generation, feature grouping will be changed and there will be a probability for every feature to be a subcomponent with interacting features. This will also increase the selection probability of a feature into the feature subset selection. RFG-2 can ensure the grouping of any number of interacting features in a subcomponent similar to RFG-1. Therefore, Theorem 1 holds not only for RFG-1, but also for RFG-2, in terms of defining the probability of grouping interacting features.

Algorithm 4 is the pseudocode of the proposed random feature grouping technique

Algorithm 4 RFG-2

Require: f, l;
1: Initialize l;
2: Apply a shuffling technique to randomize the feature indices;
3: Decompose the shuffled list of feature indices $sublists$ based on the number of features (l) in a subcomponent;
4: **return** $sublists$

(RFG-2).

RFG-3

An n-dimensional dataset is decomposed randomly into m subcomponents in every generation and all individuals have the same features in the various subcomponents in each generation. For example, if a dataset has 9 features, RFG-3 groups 9 groups into 3 subcomponents ($s = 4$) for all generations randomly and for all individuals with random features as follows:

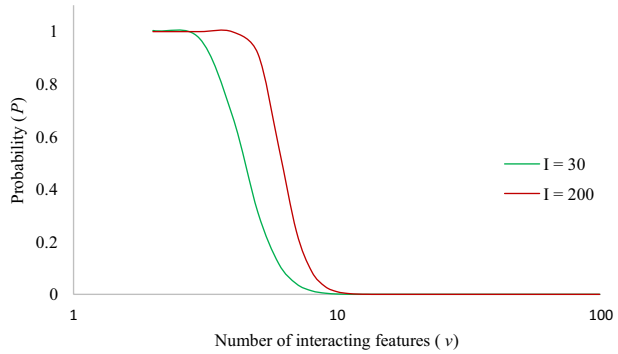

Fig. 4 Given $s = 3$, the probability of grouping interacting features is decreasing with respect to the increasing number of interacting features v over 30 individuals

$$ind_0 : \{f_2,f_7,f_1,f_5\}, \{f_3,f_4,f_6,f_9\}, \{f_8\}$$
$$ind_1 : \{f_3,f_4,f_8,f_2\}, \{f_6,f_7,f_5,f_9\}, \{f_1\}$$
$$\dots$$
$$ind_i : \{f_8,f_5,f_6,f_7\}, \{f_2,f_4,f_3,f_1\}, \{f_9\}.$$

In this method, in each generation, the grouping of features will be changed for all individuals. RFG-3 can increase the probability for every feature to be a subcomponent with interacting features and can also increase the selection probability of features into subset selection. RFG-3 can ensure the grouping of any number of interacting features in a subcomponent similar to the more frequent random grouping (DECC-ML) [14], which is often used for grouping interacting variables for function optimization problems. Equation (10) also holds for RFG-3, which is adapted from [14].

For example, for $i = 30$, $s = 3$ and $v = 4$,

$$P(F \geqslant 1) = 1 - P(F = 0) = 1 - \left(\frac{1}{3^{4-1}}\right)^{30} \approx 0.68;$$

For $i = 30$, $s = 3$ and $v = 5$,

$$P(F \geqslant 1) = 1 - P(F = 0) = 1 - \left(\frac{1}{3^{5-1}}\right)^{30} \approx 0.31.$$

This means that over 30 individuals, the probability of grouping 4 interacting features into a single subcomponent for at least one individual is about 0.68, whereas for 5 interacting features, it is approximately 0.31. This indicates that the probability of grouping interacting features will be lower if the number of interacting features increases over the number of individuals. Figure 4 shows the probability of grouping interacting features randomly into a single subcomponent with the increase in the number of interacting features. As seen in the case of RFG-1, from Fig. 4, it can be observed that if the number of interacting features is 10, the probability of grouping these features tends to be zero. However, with the same number of interacting features, the probability is increased for at least one cycle significantly with more individuals.

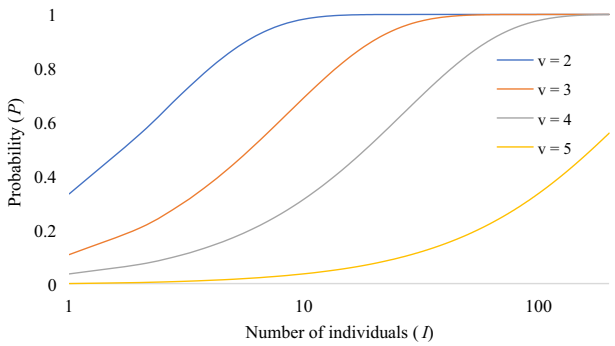

Fig. 5 Given $s = 3$, the probability of grouping interacting features is increasing with respect to the increasing number of individuals I for $v = 2, 3, 4, 5$ interacting features

Table 3 The datasets used for the experiments

Name	No. of classes	No. of features	No. of samples	Dataset domain
Breast Cancer Wsconsin[a]	2	30	569	Life (health)
Dermatology[b]	6	34	366	Life (health)
Divorce[c]	2	54	170	Life (others)
Musk[d]	2	166	6598	Physical
Pulmon[e]	4	432	58	Chemical
QSAR Oral Toxicity[f]	2	1024	8992	Physical
Colon Cancer[g]	2	2000	62	Health

[a] https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

[b] https://archive.ics.uci.edu/ml/datasets/dermatology

[c] https://archive.ics.uci.edu/ml/datasets/Divorce+Predictors+data+set

[d] https://archive.ics.uci.edu/ml/datasets/Musk+(Version+2)

[e] https://archive.ics.uci.edu/ml/datasets/Gas+sensor+array+under+flow+modulation (Feature Dataset)

[f] https://archive.ics.uci.edu/ml/datasets/QSAR+oral+toxicity

[g] http://genomics-pubs.princeton.edu/oncology/

Table 4 The CC parameters details

Phases	Options
Problem decomposition	Dynamic (RFG)
Subproblem evolution	GA
Collaboration and evaluation	$1+N$

Figure 5 illustrates the effect of increasing the number of individuals I on grouping with an increased number of interacting features into one subcomponent. It can be observed that a random grouping for every individual can increase the probability of grouping interacting features into a single subcomponent regardless the number of interacting features. For instance, for $v = 2, 3$, and 4, the highest probability of grouping features into a single subcomponent can be observed with 100 individuals.

Table 5 The decomposition parameters used for the experiments

Name	No. of subpopulation	Subpopulation size
Breast Cancer Wisconsin	3 [10, 10, 10]	30
Dermatology	3 [12, 12, 10]	30
Divorce	3 [18, 18, 18]	30
Musk	3 [56, 56, 54]	30
Pulmon	3 [144, 144, 144]	30
QSAR Oral Toxicity	4 [256, 256, 256, 256]	30
Colon Cancer	2000 [250, 250, 250, 250, 250, 250, 250, 250]	30

Table 6 The GA parameters details

Parameters	Value
Individual representation	Binary (0 or 1)
Crossover rate	100%
Mutation rate	5%
Elitism	1
Selection strategy	Tournament selection

Results and discussions

To evaluate our algorithms, experiments have been performed on seven datasets, which have been collected from the publicly available UCI machine library repository.

Dataset details

Table 3 lists the datasets used in the experiments.

The seven different datasets have been used with increasing complexities. The datasets have been selected with a dimensionality between 30 and 2000 and samples between 62 and 8992. These include datasets with a low number of features, datasets with a high number of samples, and datasets with a high number of features and datasets with a low number of samples. Dataset descriptions can be found in the UCI ML Repository and Princeton University Genomics Repository.

The CC parameters

The common CC parameters used in the experiments are listed in Table 4.

The problem decomposition parameters used in the experiments are listed in Table 5.

The common GA parameters used in the experiments are listed in Table 6.

In the case of GA optimization, the binary representation of the population is used, in which a binary 1 indicates that a feature is selected and a binary 0 indicates that a feature is not selected from the dataset. Subpopulations were initialized randomly at generation 0. For CCEAFS and CCFSRFG-1, in generation 0, since there is no previous history, to evaluate an individual in a subpopulation, random collaboration has been performed

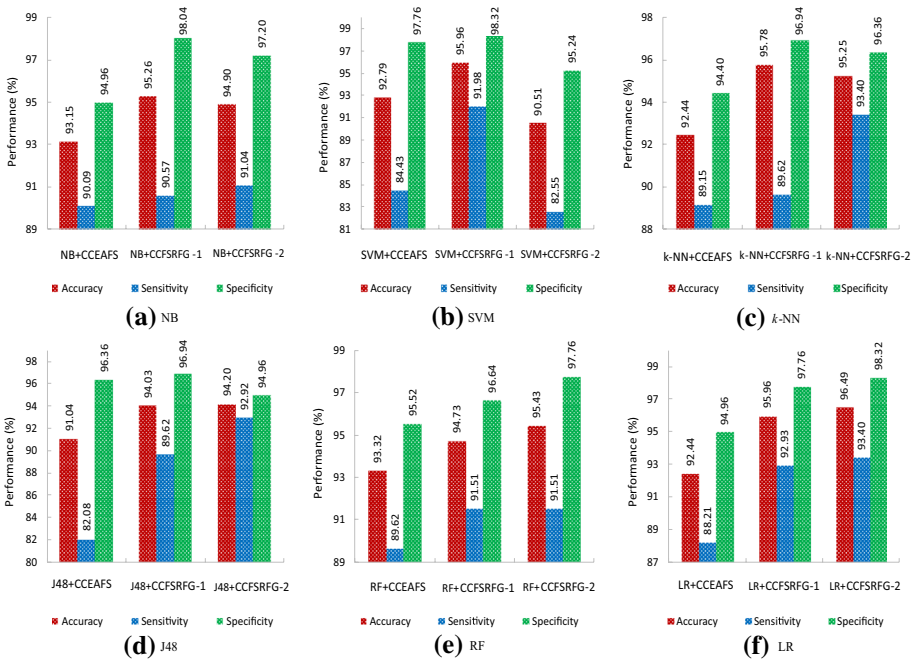

Fig. 6 Performance evaluation of classifiers on the Breast Cancer Wisconsin Dataset

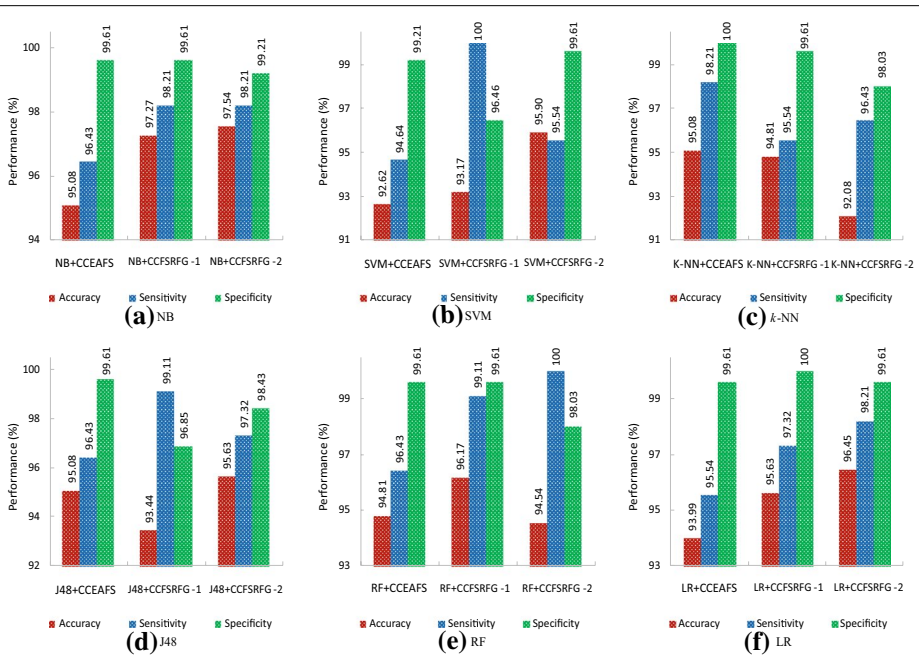

Fig. 7 Performance evaluation of classifiers on the Dermatology Dataset

to collaborate with individuals from other subpopulations. In the subsequent generations, the best individuals from the previous generation were used as the collaborators for evaluating an individual in a subpopulation. For CCFSRFG-2, in every generation, a

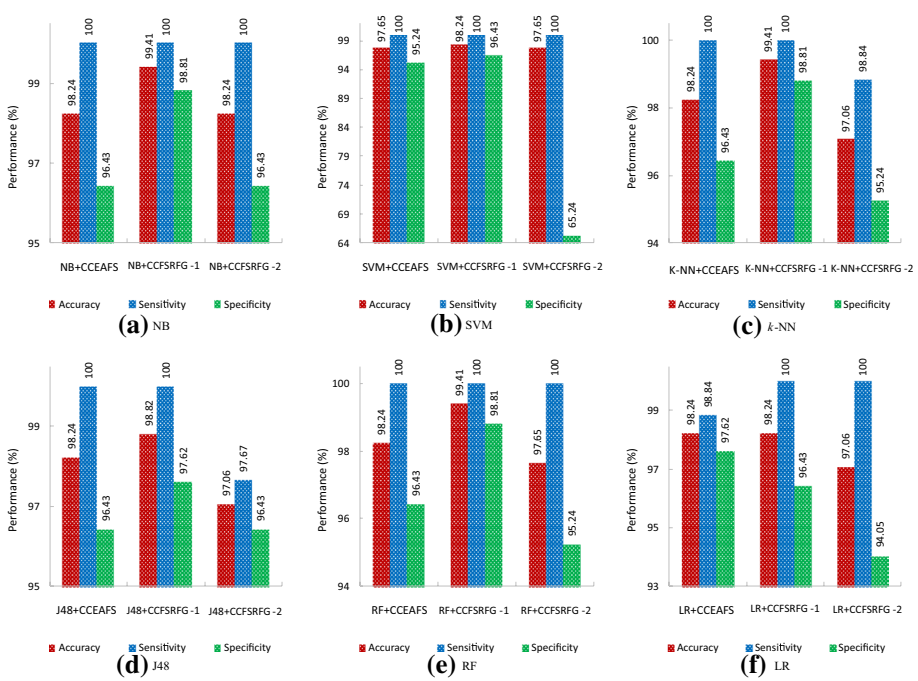

Fig. 8 Performance evaluation of classifiers on the Divorce Dataset

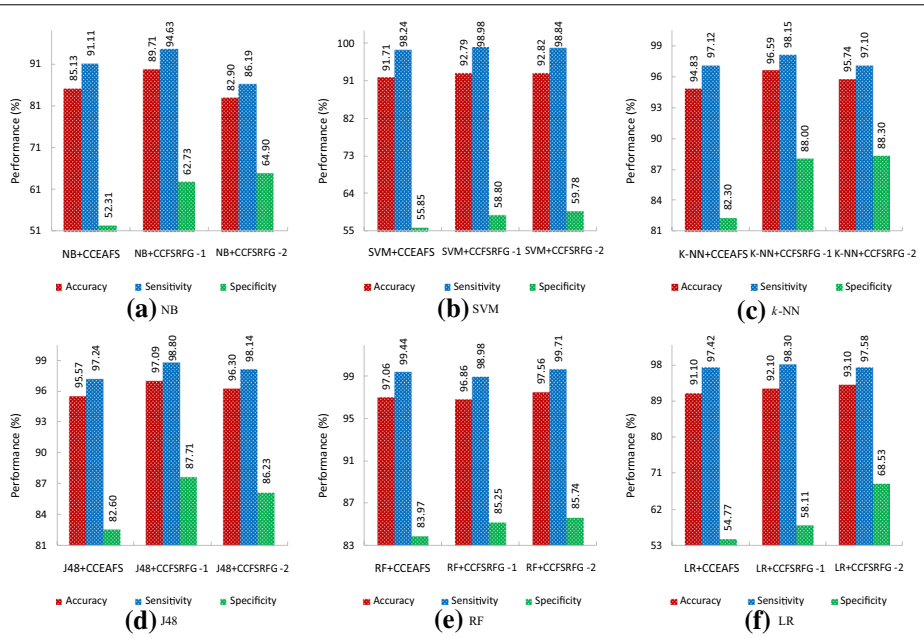

Fig. 9 Performance evaluation of classifiers on the Musk Dataset

random collaboration was performed to collaborate with individuals from other subpopulations. Collaboration performance, i.e., the fitness value was assigned to the individual being evaluated. The best individuals were combined from all subpopulations to obtain the best individual in a generation. To evaluate an individual in any subpopulation, the

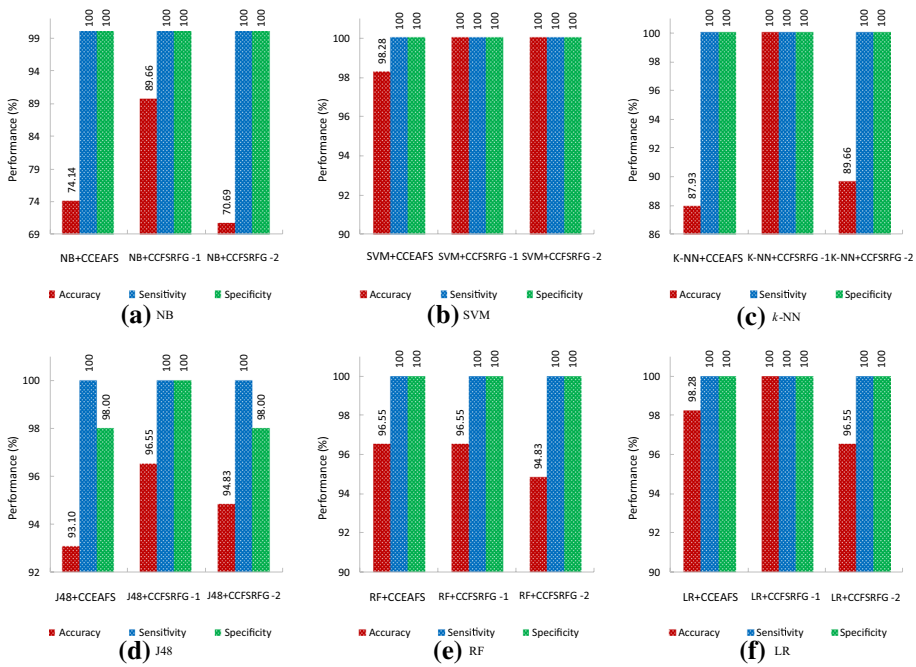

Fig. 10 Performance evaluation of classifiers on the Pulmon Dataset

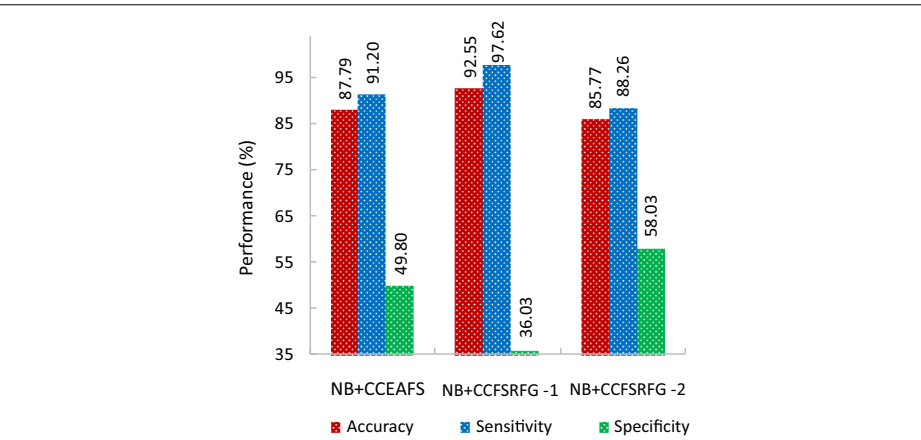

Fig. 11 Performance evaluation of the NB classifier on the QSAR Oral Toxicity Dataset

parameters for weightings w_1 and w_2 were set to 0.6 and 0.4, respectively. A maximum number of 10,000 generations were allowed as a terminating condition for the evolutionary process while verifying that there is no further improvement in the fitness value. In the case of CCEAFS, the lack of further improvement check was checked over 50 consecutive generations, and for CCFSRFG, over 100 consecutive generations.

Evaluation measures

The quality of a machine learning model is usually measured by multiple evaluation metrics. In this article, these are accuracy, sensitivity, and specificity.

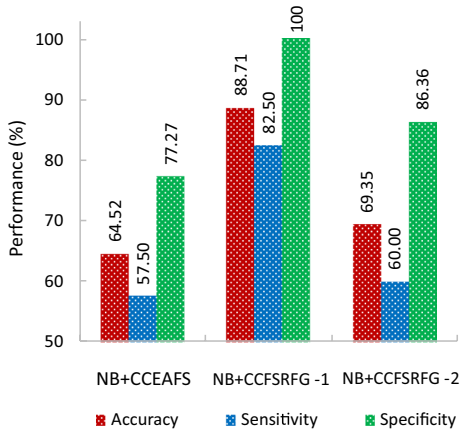

Fig. 12 Performance evaluation of the NB classifier on the Colon Cancer Dataset

Performance evaluation of classifiers with CCFSRFG

For the experiments, six widely used classifiers (NB, SVM, *k*-NN, J48, RF, and LR) have been used. First, the datasets have been tested with these classifiers without feature selection. Second, CCFSRFG-1 and CCFSRFG-2 have been used to reduce the number of features in the datasets. All six classifiers have been used to evaluate the performance of dimensionality reduction, i.e., feature selection for five datasets and because of higher computational resource requirement, only naïve Bayes has been applied to the QSAR Oral Toxicity and Colon Cancer Dataset. Figures 6, 7, 8, 9, 10, 11, 12 show the comparative performance evaluation of classifiers using FS on each dataset in terms of accuracy, sensitivity, and specificity.

Simulation results from Fig. 6 on the Wisconsin Breast Cancer Dataset show that the use of RFG in CC decomposition significantly improves the performance of both variants of the CCFSRFG framework compared to the static decomposition of CCEAFS.

It can be observed that all classifiers are equally good in terms of accuracy, sensitivity, and specificity when using CCFSRFG compared to using CCEAFS. The highest classification accuracy of 96.49% was achieved by the LR classifier when combined with CCFSRFG-2. The specificity measure was similarly better for CCFSRFG variants than the specificity obtained using CCEAFS. With the exception of SVM + CCFSRFG-2, the sensitivity obtained by the CCFSRFG variants was better than that of CCEAFS. While CCFSRFG-1 and CCFSRFG-2 both perform comparatively better than CCEAFS, CCFSRFG-1 is the better of the two in most cases.

The performance measures of all classifiers using FS on the Dermatology Dataset in Fig. 7 indicate that with the exception of the *k*-NN and J48 classifiers, all classifiers perform better with RFG variants than the static decomposition of CCEAFS.

The classification accuracy for this dataset was obtained by NB + CCFSRFG-2, which is 97.54%. With *k*-NN + CCFSRFG-2, a 3% reduction in accuracy was observed, and a similar reduction (about 2%) was noted with J48 + CCFSRFG-1. A few cases, for example, SVM + CCFSRFG-1 and RF + CCFSRFG-1, resulted in a 100% sensitivity. In

the case of *k*-NN + CCFSRFG-2 and LR + CCFSRFG-2, not only sensitivity was 100%, but also specificity.

According to the performance results of all classifiers with an FS on Divorce Dataset (see Fig. 8), it can be seen that in terms of accuracy, NB, SVM, *k*-NN, J48, and RF in collaboration with CCFSRFG-1 perform better than the other variant and CCEAFS.

In the case of LR, it equally performs good in terms of accuracy using CCEAFS and CCFSRFG-1. The highest classification accuracy achieved by most of the classifiers was almost 100%. In a few cases, for example, NB and SVM, the same accuracy was achieved by CCEAFS and CCFSRFG-2. It is clear that except *k*-NN + CCFSRFG-2, J48 + CCFSRFG-2, and LR + CCEAFS, a 100% sensitivity was achieved by the feature selection process. However, the specificity achieved is smaller compared to other performance measures, which is especially true for SVM + CCFSRFG-2 (65.24%).

Figure 9 illustrates the performance results of all classifiers with feature selection on Musk Dataset.

From this simulation, it can be observed that the highest classification has been obtained by the RF classifier when applied with CCFSRFG-1 (97.56%). In terms of static decomposition (CCEAFS) and dynamic decomposition (CCFS with a variant of RFG), it can be seen that in most cases, the dynamic decomposition perform better than the static decomposition in all measures, except two cases in which the performance measures have not been reduced significantly (e.g., NB + CCFSRFG-2 and RF + CCFSRFG-1). Apart from classification accuracy, the sensitivity achieved by all classifiers was above 90%; however, specificity was only between 54.77 and 88.30%.

The performance evaluation results of classifiers on the Pulmon Dataset are displayed in Fig. 10.

The figure shows that a 100% classification can be achieved by SVM, *k*-NN, and LR classifiers when they are used with a dynamic decomposition (RFG-1 or RFG-2) technique. In all cases, CCFSRFG-1 performs better than CCEAFS except by RF where it equally performs good with CCEAFS. However, CCFSRFG-2 performs equally better only in the case of the SVM classifier. In a few cases, CCFSRFG-2 performs slightly better than CCEAFS. With the exception with J48 classifier, in all other cases, the sensitivity and the specificity obtained by each classifier were 100%.

The NB classifier performance on the QSAR Oral Toxicity Dataset in Fig. 11 shows that in terms of accuracy and sensitivity, all feature selection approaches are equally good. However, it also shows that the RFG-1 variant in the CCFS framework performs better in terms of the aforementioned measures. In contrast, the CCFSRFG-1 results in a much lower specificity compared to CCEAFS and CCFSRFG-2. The highest accuracy, sensitivity, and specificity achieved by the NB classifier on this dataset were 92.55%, 97.62%, and 58.03%, respectively.

Figure 12 shows the performance of the NB classifier on the Colon Cancer Dataset in terms of accuracy, sensitivity, and specificity.

In this case, the random grouping with its two variants in CCFS framework perform equally better than the static decomposition in the CCEAFS framework. Within the CCFSRFG variants, CCFS with RFG-1 performs much better than the CCFS with RFG-2. The accuracy, sensitivity, and specificity obtained by CCFSRFG-1 were 88.71%, 82.50%, and 100%, respectively.

Table 7 Summary of results for the Wisconsin Breast Cancer Dataset

Classifier	Accuracy (%)	Sensitivity (%)	Specificity (%)	No. of features
NB	92.79	89.15	94.96	30
NB + CCEAFS	93.15	90.09	94.96	*2*
NB + CCFSRFG-1	*95.26*	*90.57*	*98.04*	*2*
NB + CCFSRFG-2	94.90	91.04	97.20	6
SVM	*97.72*	*94.81*	*99.44*	30
SVM + CCEAFS	92.79	84.43	97.76	*3*
SVM + CCFSRFG-1	95.96	91.98	98.32	*3*
SVM + CCFSRFG-2	90.51	82.55	95.24	*3*
k-NN	95.43	*94.34*	96.08	30
k-NN + CCEAFS	92.44	89.15	94.40	*3*
k-NN + CCFSRFG-1	*95.78*	89.62	*96.94*	*3*
k-NN + CCFSRFG-2	95.25	*93.40*	96.36	7
J48	92.97	91.04	94.12	30
J48 + CCEAFS	91.04	82.08	96.36	*1*
J48 + CCFSRFG-1	94.03	89.62	*96.94*	2
J48 + CCFSRFG-2	*94.20*	*92.92*	94.96	5
RF	*96.13*	*93.40*	*97.76*	30
RF + CCEAFS	93.32	89.62	95.52	*2*
RF + CCFSRFG-1	94.73	*91.51*	96.64	*2*
RF + CCFSRFG-2	*95.43*	*91.51*	*97.76*	6
LR	94.90	*94.81*	94.96	30
LR + CCEAFS	92.44	88.21	94.96	*2*
LR + CCFSRFG-1	95.96	92.93	97.76	*2*
LR + CCFSRFG-2	*96.49*	*93.40*	*98.32*	6

Italic signifies the improvements of the proposed approach over existing techniques

The experimental results for all of the datasets used in this paper are summarized in Tables 7, 8, 9, 10, 11, 12, 13.

Table 7 shows the summary results for the Wisconsin breast cancer dataset.

The table indicates that all classifiers are equally good in terms of accuracy, sensitivity, and specificity measures when they are used with CC-based FS framework with RFG-1 and RFG-2, with an exception of SVM, the classifier performs better without using feature selection. It can be observed that the feature reduction rate varies among the classifiers. For example, both CCEAFS and CCFSRFG-1 reduce features in the dataset by 93.33% with the NB, RF, and LR classifiers, and 90% with SVM and *k*-NN. Although J48 can reduce features in the dataset by 96.67% when combined with CCEAFS, a higher performance can be achieved when combined with CCFS and RFG variants. In most cases, CCFSRFG-2 achieves higher performance results, however, the feature reduction rate is not lower compared to the other variant and CCEAFS.

The summary of results for the Dermatology Dataset is listed in Table 8.

From the table, it can be observed that all classifiers are equally good in terms of accuracy, sensitivity, and specificity, regardless whether they are used with or without CCFS. However, when the classifiers are used with CCFS, a feature reduction rate between 70.59 and 85.29% can be achieved. CCFSRFG-1 reduces the number of

Table 8 Summary of results for the Dermatology Dataset

Classifier	Accuracy (%)	Sensitivity (%)	Specificity (%)	No. of features
NB	*97.54*	*99.11*	*100.00*	34
NB + CCEAFS	95.08	96.43	99.61	7
NB + CCFSRFG-1	97.27	98.21	99.61	*6*
NB + CCFSRFG-2	*97.54*	98.21	99.61	7
SVM	*96.99*	*100.00*	*100.00*	34
SVM + CCEAFS	92.62	94.64	99.21	8
SVM + CCFSRFG-1	93.17	*100.00*	96.46	*5*
SVM + CCFSRFG-2	95.90	95.54	99.61	10
k-NN	*95.63*	*99.11*	*100.00*	34
k-NN + CCEAFS	95.08	98.21	*100.00*	8
k-NN + CCFSRFG-1	94.81	95.54	99.61	*5*
k-NN + CCFSRFG-2	92.08	96.43	98.03	9
J48	*95.90*	97.32	*100.00*	34
J48 + CCEAFS	95.08	96.43	99.61	7
J48 + CCFSRFG-1	93.44	*99.11*	96.85	*5*
J48 + CCFSRFG-2	95.63	97.32	98.43	9
RF	*97.54*	*100.00*	*99.61*	34
RF + CCEAFS	94.81	96.43	*99.61*	7
RF + CCFSRFG-1	96.17	99.11	*99.61*	*6*
RF + CCFSRFG-2	94.54	*100.00*	98.03	10
LR	*96.99*	*100.00*	99.61	34
LR + CCEAFS	93.99	95.54	99.61	7
LR + CCFSRFG-1	95.63	97.32	*100.00*	*5*
LR + CCFSRFG-2	96.45	98.21	99.61	8

Italic signifies the improvements of the proposed approach over existing techniques

features by 1 to 3 features; however, CCFSRFG-2 cannot reduce the number of feature significantly to CCEAFS.

From the summary results of the Divorce Dataset in Table 9, it can be observed that all classifiers result in a higher performance in terms of accuracy, sensitivity, and specificity when used with CCFSRFG-1, and the number of features is reduced to only 2, except for SVM, where the number of features is 3. The rate of feature reduction varies between 75.93 and 96.30%.

The summary of results for the Must Dataset in Table 10 indicates that in terms of accuracy, sensitivity, and specificity, NB, *k*-NN, and J48 can achieve the highest classification accuracy with a much reduced number of features when combined with CCFSRFG-1.

The other classifiers can also reduce the number of features to a very low level without significantly reducing classification accuracy and other measures. The maximum number of features can be reduced by RF + CCFSRFG-1 (from 166 to only 7) with a reduction rate of 95.78%. Overall, the feature reduction rate by the classifiers varies from 87.95 to 95.78%, when they are used with a feature selection process with RFG-1, which clearly indicates superior performance over CCEAFS approaches. While the number of feature reduction by CCFSRFG-2 lies between 56 to 65, classification

Table 9 Summary of results for the Divorce Dataset

Classifier	Accuracy (%)	Sensitivity (%)	Specificity (%)	No. of features
NB	97.65	100.00	95.24	54
NB + CCEAFS	98.24	100.00	96.43	3
NB + CCFSRFG-1	*99.41*	100.00	*98.81*	*2*
NB + CCFSRFG-2	98.24	100.00	96.43	12
SVM	*98.24*	100.00	*96.43*	54
SVM + CCEAFS	97.65	100.00	95.24	3
SVM + CCFSRFG-1	*98.24*	100.00	*96.43*	*2*
SVM + CCFSRFG-2	97.65	100.00	95.24	9
k-NN	97.65	100.00	95.24	54
k-NN + CCEAFS	98.24	100.00	96.43	3
k-NN + CCFSRFG-1	*99.41*	100.00	*98.81*	*2*
k-NN + CCFSRFG-2	97.06	98.84	95.24	13
J48	97.06	97.67	96.43	54
J48 + CCEAFS	98.24	*100.00*	96.43	3
J48 + CCFSRFG-1	*98.82*	*100.00*	*97.62*	*2*
J48 + CCFSRFG-2	97.06	97.67	96.43	11
RF	97.65	100.00	95.24	54
RF + CCEAFS	98.24	100.00	96.43	*3*
RF + CCFSRFG-1	*99.41*	100.00	*98.81*	*3*
RF + CCFSRFG-2	97.65	100.00	95.24	13
LR	97.65	*100.00*	95.24	54
LR + CCEAFS	*98.24*	98.84	*97.62*	3
LR + CCFSRFG-1	*98.24*	*100.00*	96.43	*2*
LR + CCFSRFG-2	97.06	*100.00*	94.05	13

Italic signifies the improvements of the proposed approach over existing techniques

accuracy is higher than CCEAFS in most cases, although at the cost of more number of features for CCFSRFG-2 compared to CCEAFS.

When the dataset has a large number of features and a low number of samples, as in the case of Pulmon Dataset, it can be seen from the summary results in Table 11 that CCFSRFG-1 can reduce the number of features much more than CCEAFS.

k-NN + CCFSRFG-1 can reduce the number of features from 432 to only 25 (feature reduction rate of 94.21%) with a 100% accuracy. This accuracy can also be achieved by SVM + CCFSRFG-1, SVM + CCFSRFG-2, and LR + CCFSRFG-1; however, the feature reduction rate is less than that of *k*-NN + CCFSRFG-1. With the exception by J48, all classifiers achieved a 100% sensitivity and specificity.

With a large number of features (1024) in QSAR Oral Toxicity Dataset, it can be seen from the summary results in Table 12 that the NB classifier can reduce the number of features substantially (to only 60) when NB is combined with CCFSRFG-1.

With this framework, NB also can obtain a higher classification rate of 92.55% compared to the 79.78% of NB without using any feature selection. NB can reduce the number of features to a very lower number whether used with a CC-based FS framework using static decomposition method or with a dynamic decomposition method, such as RFG. However, it can be clearly predicted that NB performs very well when it is used with CCFSRFG-1, compared to other feature selection combinations.

Table 10 Summary of results for the Musk Dataset

Classifier	Accuracy (%)	Sensitivity (%)	Specificity (%)	No. of features
NB	84.04	85.88	*73.94*	166
NB + CCEAFS	85.13	91.11	52.31	25
NB + CCFSRFG-1	*89.71*	*94.63*	62.73	*20*
NB + CCFSRFG-2	82.90	86.19	64.90	56
SVM	*94.82*	*99.05*	*71.58*	166
SVM + CCEAFS	91.71	98.24	55.85	22
SVM + CCFSRFG-1	92.79	98.98	58.80	*14*
SVM + CCFSRFG-2	92.82	98.84	59.78	63
k-NN	95.76	97.12	*88.30*	166
k-NN + CCEAFS	94.83	97.12	82.30	18
k-NN + CCFSRFG-1	*96.59*	*98.15*	88.00	*9*
k-NN + CCFSRFG-2	95.74	97.10	*88.30*	61
J48	96.85	98.15	*89.68*	166
J48 + CCEAFS	95.57	97.94	82.60	14
J48 + CCFSRFG-1	*97.09*	*98.80*	87.71	*9*
J48 + CCFSRFG-2	96.30	98.14	86.23	65
RF	*97.94*	*99.87*	*87.32*	166
RF + CCEAFS	97.06	99.44	83.97	16
RF + CCFSRFG-1	96.86	98.98	85.25	*7*
RF + CCFSRFG-2	97.56	99.71	85.74	60
LR	*95.29*	97.94	*80.83*	166
LR + CCEAFS	91.10	97.42	54.77	23
LR + CCFSRFG-1	92.10	*98.30*	58.11	*10*
LR + CCFSRFG-2	93.10	97.58	68.53	62

Italic signifies the improvements of the proposed approach over existing techniques

The summary results of the Colon Cancer Dataset are recorded in Table 13.

Similar to the QSAR Oral Toxicity Dataset, the Colon Cancer Dataset also has a large number of features. This is why NB + CCFSRFG-1 performs much better in terms of accuracy, sensitivity, and specificity than NB + CCEAFS or NB + CCFSRFG-2. A 96.35% feature reduction can be observed for NB + CCFSRFG-1 on the Colon Cancer Dataset.

According to the experiments, when the dimensionality of a dataset is reduced using feature selection, the performance measures in terms of accuracy, sensitivity, and specificity are not degraded in a significant rate. In a few cases, it can be observed that the classifiers perform better with feature selection than without it. Because of the search algorithm, the selected subset of features is not unique for every classifier. If the subset of features evaluated by two different classifiers is the same, the underlying search algorithm returns the same subset of features with the two independent executions. The classifiers are only used to evaluate the selected subset of features, thereby improving classification accuracy in each generation. It can also be seen that random feature grouping (RFG) introduced in this paper significantly improves classification performance. As studied in the literature, decomposition strategy plays an important role in the performance of a CC-based framework. The experimental results obtained here confirm the role of dynamic decomposition methods. In this paper, three variants of RFG have been proposed, the first two of which tested using all seven datasets. The second variant, RFG-2, which allows a dataset to be decomposed in every generation of an evolutionary

Table 11 Summary of results for the Pulmon Dataset

Classifier	Accuracy (%)	Sensitivity (%)	Specificity (%)	No. of features
NB	65.52	100.00	100.00	432
NB + CCEAFS	74.14	100.00	100.00	91
NB + CCFSRFG-1	*89.66*	100.00	100.00	*42*
NB + CCFSRFG-2	70.69	100.00	100.00	208
SVM	93.10	100.00	100.00	432
SVM + CCEAFS	98.28	100.00	100.00	89
SVM + CCFSRFG-1	*100.00*	100.00	100.00	*51*
SVM + CCFSRFG-2	*100.00*	100.00	100.00	190
k-NN	86.21	100.00	100.00	432
k-NN + CCEAFS	87.93	100.00	100.00	105
k-NN + CCFSRFG-1	*100.00*	100.00	100.00	*25*
k-NN + CCFSRFG-2	89.66	100.00	100.00	188
J48	82.76	62.50	96.00	432
J48 + CCEAFS	93.10	*100.00*	98.00	61
J48 + CCFSRFG-1	*96.55*	*100.00*	96.55	*29*
J48 + CCFSRFG-2	94.83	*100.00*	98.00	193
RF	91.38	100.00	100.00	432
RF + CCEAFS	*96.55*	100.00	100.00	*117*
RF + CCFSRFG-1	*96.55*	100.00	100.00	183
RF + CCFSRFG-2	94.83	100.00	100.00	210
LR	86.21	100.00	100.00	432
LR + CCEAFS	98.28	100.00	100.00	143
LR + CCFSRFG-1	*100.00*	100.00	100.00	*137*
LR + CCFSRFG-2	96.55	100.00	100.00	219

Italic signifies the improvements of the proposed approach over existing techniques

Table 12 Summary of results for the QSAR Oral Toxicity Dataset

Classifier	Accuracy (%)	Sensitivity (%)	Specificity (%)	No. of features
NB	79.78	80.79	*68.56*	1024
NB + CCEAFS	87.79	91.20	49.80	201
NB + CCFSRFG-1	*92.55*	*97.62*	36.03	*60*
NB + CCFSRFG-2	85.77	88.26	58.03	467

Italic signifies the improvements of the proposed approach over existing techniques

Table 13 Summary of results for the Colon Cancer Dataset

Classifier	Accuracy (%)	Sensitivity (%)	Specificity (%)	No. of features
NB	58.06	50.00	72.73	2000
NB + CCEAFS	64.52	57.50	77.27	469
NB + CCFSRFG-1	*88.71*	*82.50*	*100.00*	*73*
NB + CCFSRFG-2	69.35	60.00	86.36	1004

Italic signifies the improvements of the proposed approach over existing techniques

process, achieved a higher classification performance than RFG-1, in case of which the dataset is decomposed randomly only once (at the beginning of an evolutionary process). RFG-2 can ensure the grouping of interacting features in the same subcomponent; however, it fails to reduce the number of features significantly (compared to RFG-1).

Although RFG-2 achieves a higher classification performance in a few cases this comes at a cost of more features than with RFG-1 or CCEAFS. RFG-2 could not perform better than RFG-1 and static decomposition because of the collaboration technique used (i.e., an individual from a subpopulation with the collaboration of individuals from the rest of each subpopulation) for the problem. In the case of CCEAFS and CCFSRFG-1, in the very first generation (where there is no previous information available), a random collaboration technique can be used to perform the collaboration. From the second generation onwards, the best individuals at the time can be used from the previous generation to perform the collaboration for all individuals. However, in the case of CCFSRFG-2, for every generation, only a random collaboration technique is used to perform the collaboration in every generation. The best individuals at the time cannot be used from the previous generation as collaborators for subsequent generations for CCFSRFG-2 because of the random group components in every generation. Random group components (i.e., the features in a subpopulation) are usually different from the group components of previous generations. Hence, if the best individual from a previous generation collaborates with an individual in the current generation, the same features might collaborate twice or multiple times, which results in a contradictory situation, because the features represented as both 0 and 1 would indicate not being selected and being selected at the same time in the subset. Thus, when the same feature is included in collaborating individuals twice or multiple times with the same or different representation, a formation of a complete solution is not possible. This concludes that beyond an appropriate decomposition method, a suitable collaboration technique is also needed for a CC-based FS framework, which is yet to be investigated. Alhough RFG-3 was proposed as a third variant of random feature grouping this also relies on a proper collaboration technique.

Feature selection is an essential preprocessing step in various Big Data analytics, for example, anomaly detection for cybersecurity data. Cybersecurity data analysis is very important to indicate vulnerabilities and unveil security breaches, such as via detecting network inconsistencies. Examples of applying feature selection before anomaly detection include a hierarchical feature selection for DDoS mitigation [71], an ensemble feature selection for intrusion detection [72], and clustering and correlation-based feature selection for intrusion detection [73]. Hence, our proposed cooperative co-evolution-based feature selection with the proposed random feature grouping (CCFSRFG) can be applied in any domain of Big Data.

Conclusion and future work

Following an extensive literature review on problem decomposition approaches using CC, this paper investigated the application of CC with a dynamic decomposition for feature selection. It proposed a random feature grouping algorithm for feature selection using CC, and evaluated its performance of six ML classifiers on seven datasets. The experiments indicated that the feature selection process does not degrade the performance of the classifiers significantly. We also analyzed the effect of feature selection on different datasets, covering datasets with many samples and few features, as well as datasets with few samples and many features.

The comparative analysis of the performance results of the classifiers in terms of accuracy, sensitivity, and specificity validate the effectiveness of the proposed CC-based

feature selection algorithm (CCFSRFG) with the novel dynamic decomposition technique, random feature grouping (RFG), for feature selection problems. In most cases, the proposed FS approach with RFG-1 results in a better performance than the static decomposition approach (CCEAFS) and when using without feature selection. Although a suitable problem decomposition technique for feature selection may result in better performance, a suitable decomposition strategy by itself is not sufficient to achieve the full potential of CC-based approaches. CC-based approaches also require an appropriate optimizer to evolve each subpopulation and a proper collaboration technique to build a complete solution. For example, CCFSRFG-2 may perform better with efficient feature reduction when using other collaboration model. Therefore, as future work, a proper collaboration model will be investigated for the proposed CCFSRFG approach. In addition, CCFSRFG will be combined with other optimizers.

Abbreviations

CC: Cooperative co-evolution; CCEAFS: Cooperative co-evolutionary algorithm based feature selection; CCFSRFG: Cooperative co-evolutionary-based feature selection with random feature grouping; EA: Evolutionary algorithm; FS: Feature selection; IoT: Internet of Things; *k*-NN: *k*-Nearest Neighbor; NB: Naïve Bayes; ML: Machine learning; RF: Random forest; RFG: Random feature grouping; RG: Random grouping; SVM: Support vector machine.

Acknowledgements

This research is supported by the Edith Cowan University (ECU) Higher Degree by Research Scholarship (HDRS) and the ECU School of Science Research Scholarship. The authors would like to thank Dr. Erchuan Zhang of the ECU School of Science to assist in defining the probability functions for the proposed random feature grouping decomposition strategies.

Authors' contributions

BR conceived of the presented idea, developed the theory, implemented the research, and wrote the manuscript. BR, MA, and LS contributed to the design, and to the analysis of the results. All authors reviewed and approved the final manuscript.

Authors' information

A. N. M. Bazlur Rashid: Since 2012, he has been an assistant professor (Computer) at Bangladesh University of Textiles. He is the author of a number of journal articles and conference papers. His research interests include evolutionary computation, machine learning, big data optimization, data science, knowledge discovery, and decision support systems.

M. Ahmed: His research intersects between data analytics and cybersecurity. He is a Senior Member of IEEE and member of Australian Computer Society/certified professional.

Leslie F. Sikos, Ph.D., is a computer scientist specializing in network forensics and cybersecurity applications powered by artificial intelligence and data science. He has industry experience in data center and cloud infrastructures, cyberthreat prevention and mitigation, and firewall management. Dr. Sikos holds professional certificates, and is a member of the IEEE Computer Society Technical Committee on Security and Privacy, and a founding member of the IEEE Special Interest Group on Big Data for Cybersecurity and Privacy.

Paul Haskell-Dowland is an Associate Professor, and the Associate Dean for Computing and Security in the School of Science at Edith Cowan University and is an associate member of the Centre for Security, Communications & Network Research at Plymouth University (UK). Paul has delivered keynotes, invited presentations, workshops, professional development/training and seminars across the world for audiences including RSA Security, Sri Lanka CERT, ITU and IEEE. He has appeared on local and national media (newspaper, radio and tv) commenting on current cyber issues as well as contributions through articles published in The Conversation. Paul has more than 20 years of experience in cyber security research and education in both the UK and Australia.

Funding

This work was not funded.

Availability of data and materials

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Received: 7 August 2020 Accepted: 15 November 2020
Published online: 04 December 2020

References

1. Rashid ANMB. Access methods for Big Data: current status and future directions. EAI Endorsed Trans Scalable Inf Syst. 2018. https://doi.org/10.4108/eai.28-12-2017.153520.
2. Chakraborty B, Kawamura A. A new penalty-based wrapper fitness function for feature subset selection with evolutionary algorithms. J Inf Telecommun. 2018;2(2):163–80. https://doi.org/10.1080/24751839.2018.1423792.
3. Khalid S, Khalil T, Nasreen S. A survey of feature selection and feature extraction techniques in machine learning. In: 2014 science and information conference. 2014. p. 372–8 . https://doi.org/10.1109/SAI.2014.6918213.
4. Miao J, Niu L. A survey on feature selection. Procedia Comput Sci. 2016;91:919–26. https://doi.org/10.1016/j.procs.2016.07.111.
5. Rashid AB, Choudhury T. Knowledge management overview of feature selection problem in high-dimensional financial data: cooperative co-evolution and MapReduce perspectives. Probl Perspect Manag. 2019;17(4):340. https://doi.org/10.21511/ppm.17(4).2019.28.
6. Liu Y, Tang F, Zeng Z. Feature selection based on dependency margin. IEEE Transn Cybern. 2014;45(6):1209–21. https://doi.org/10.1109/TCYB.2014.2347372.
7. Stanovov V, Brester C, Kolehmainen M, Semenkina O. Why don't you use evolutionary algorithms in big data? In: IOP conference series: materials science and engineering. Bristol: IOP Publishing; 2017. vol. 173, p. 012020. https://doi.org/10.1088/1757-899x/173/1/012020.
8. Wang R, Zhang F, Zhang T, Fleming PJ. Cooperative co-evolution with improved differential grouping method for large-scale global optimisation. Int J Bio-Inspired Comput. 2018;12(4):214–25.
9. Sun Y, Kirley M, Halgamuge SK. On the selection of decomposition methods for large scale fully non-separable problems. In: Proceedings of the companion publication of the 2015 annual conference on genetic and evolutionary computation. New York: ACM; 2015. p. 1213–6. https://doi.org/10.1145/2739482.2768483.
10. Sun Y, Kirley M, Halgamuge SK. A recursive decomposition method for large scale continuous optimization. IEEE Trans Evol Comput. 2018;22(5):647–61. https://doi.org/10.1109/TEVC.2017.2778089.
11. Hu X-M, He F-L, Chen W-N, Zhang J. Cooperation coevolution with fast interdependency identification for large scale optimization. Inf Sci. 2017;381:142–60. https://doi.org/10.1016/j.ins.2016.11.013.
12. Chen A, Ren Z, Yang Y, Liang Y, Pang B. A historical interdependency based differential grouping algorithm for large scale global optimization. In: Proceedings of the genetic and evolutionary computation conference companion. New York: ACM; 2018. p. 1711–5. https://doi.org/10.1145/3205651.3208278.
13. Li R, Zhang W, Zhao Y, Zhu Z, Ji S. Sparsity learning formulations for mining time-varying data. IEEE Trans Knowl Data Eng. 2015;27(5):1411–23. https://doi.org/10.1109/TKDE.2014.2373411.
14. Omidvar MN, Li X, Yang Z, Yao X. Cooperative co-evolution for large scale optimization through more frequent random grouping. In: IEEE congress on evolutionary computation. New York: IEEE; 2010. p. 1–8 . https://doi.org/10.1109/CEC.2010.5586127.
15. Ling Y, Li H, Cao B. Cooperative co-evolution with graph-based differential grouping for large scale global optimization. In: Li M, Xiong N, Tong Z, Du J, Liu C, Li K, Wang L, editors. 12th international conference on natural computation, fuzzy systems and knowledge discovery. New York: IEEE; 2016. p. 95–102. https://doi.org/10.1109/FSKD.2016.7603157.
16. Kamkar I, Gupta SK, Phung D, Venkatesh S. Stabilizing l1-norm prediction models by supervised feature grouping. J Biomed Inform. 2016;59:149–68. https://doi.org/10.1016/j.jbi.2015.11.012.
17. García-Torres M, Gómez-Vela F, Melián-Batista B, Moreno-Vega JM. High-dimensional feature selection via feature grouping: a variable neighborhood search approach. Inf Sci. 2016;326(C):102–18. https://doi.org/10.1016/j.ins.2015.07.041.
18. Gan G, Ng MKP. Subspace clustering with automatic feature grouping. Pattern Recognit. 2015;48(11):3703–13. https://doi.org/10.1016/j.patcog.2015.05.016.
19. Rashid ANMB, Ahmed M, Sikos LF, Haskell-Dowland P. A novel penalty-based wrapper objective function for feature selection in big data using cooperative co-evolution. IEEE Access. 2020;8:150113–29. https://doi.org/10.1109/ACCESS.2020.3016679.
20. Gao W, Hu L, Zhang P. Feature redundancy term variation for mutual information-based feature selection. Appl Intell. 2020. https://doi.org/10.1007/s10489-019-01597-z.
21. Guo Y, Cao X, Xu Y, Hong Q. Co-evolution based feature selection for pedestrian detection. In: 2007 IEEE international conference on control and automation. New York: IEEE; 2007. p. 2797–801 . https://doi.org/10.1109/ICCA.2007.4376871.
22. Cao X, Xu Y, Wei C, Guo Y. Feature subset selection based on co-evolution for pedestrian detection. Trans Inst Meas Control. 2011;33(7):867–79. https://doi.org/10.1177/0142331209103041.
23. Derrac J, García S, Herrera F. A first study on the use of coevolutionary algorithms for instance and feature selection. In: Corchado E, Wu X, Oja E, Herrero Á, Baruque B, editors. Hybrid artificial intelligence systems. Heidelberg: Springer; 2009. p. 557–564. https://doi.org/10.1007/978-3-642-02319-4_67.
24. Derrac J, García S, Herrera F. IFS-CoCo: instance and feature selection based on cooperative coevolution with nearest neighbor rule. Pattern Recognit. 2010;43(6):2082–105. https://doi.org/10.1016/j.patcog.2009.12.012.
25. Tian J, Li M, Chen F. Dual-population based coevolutionary algorithm for designing RBFNN with feature selection. Expert Syst Appl. 2010;37(10):6904–18. https://doi.org/10.1016/j.eswa.2010.03.031.
26. Wen Y, Xu H. A cooperative coevolution-based Pittsburgh learning classifier system embedded with memetic feature selection. In: 2011 IEEE congress of evolutionary computation. New York: IEEE; 2011. p. 2415–22. https://doi.org/10.1109/CEC.2011.5949916.
27. Ebrahimpour MK, Nezamabadi-Pour H, Eftekhari M. CCFS: a cooperating coevolution technique for large scale feature selection on microarray datasets. Comput Biol Chem. 2018;73:171–8. https://doi.org/10.1016/j.compbiolchem.2018.02.006.
28. Christo VE, Nehemiah HK, Brighty J, Kannan A. Feature selection and instance selection from clinical datasets using co-operative co-evolution and classification using random forest. IETE J Res. 2020. https://doi.org/10.1080/03772063.2020.1713917.

29. Wang Y, Qu B, Liang J, Wei Y, Yue C, Hu Y, Song H. Two-stage decomposition method based on cooperation coevolution for feature selection on high-dimensional classification. IEEE Access. 2019;7:163191–201. https://doi.org/10.1109/ACCESS.2019.2946649.
30. Shi M, Gao S. Reference sharing: a new collaboration model for cooperative coevolution. J Heuristics. 2017;23(1):1–30. https://doi.org/10.1007/s10732-016-9322-9.
31. Chen W, Weise T, Yang Z, Tang K. Large-scale global optimization using cooperative coevolution with variable interaction learning. In: Schaefer R, Cotta C, Kołodziej J, Rudolph G, editors. International conference on parallel problem solving from nature—PPSN XI. Heidelberg: Springer; 2010. p. 300–9. https://doi.org/10.1007/978-3-642-15871-1_31.
32. Chandra R, Deo R, Bali K, Sharma A. On the relationship of degree of separability with depth of evolution in decomposition for cooperative coevolution. In: 2016 IEEE congress on evolutionary computation. New York: IEEE; 2016. p. 4823–30 . https://doi.org/10.1109/CEC.2016.7744408.
33. Li X, Tang K, Omidvar MN, Yang Z, Qin K, China H. Benchmark functions for the CEC 2013 special session and competition on large-scale global optimization. Gene. 2013;7(33):8.
34. Potter MA, De Jong KA. A cooperative coevolutionary approach to function optimization. In: International conference on parallel problem solving from nature. Berlin: Springer; 1994. p. 249–57 . https://doi.org/10.1007/3-540-58484-6_269.
35. Omidvar MN, Li X. Evolutionary large-scale global optimization: An introduction. In: Proceedings of the genetic and evolutionary computation conference companion, GECCO-17. New York: ACM; 2017. p. 807–27. https://doi.org/10.1145/3067695.3067706.
36. Durand N, Alliot J-M. Genetic crossover operator for partially separable functions. In: 3rd annual conference on Genetic Programming, 1998, Madison, United States—HAL. 1998.
37. Potter MA, Jong KAD. Cooperative coevolution: an architecture for evolving coadapted subcomponents. Evol Comput. 2000;8(1):1–29. https://doi.org/10.1162/106365600568086.
38. Shi Y-J, Teng H-F, Li Z-Q. Cooperative co-evolutionary differential evolution for function optimization. In: Wang L, Chen K, Ong YS, editors. Advances in natural computation. Heidelberg: Springer; 2005. p. 1080–1088. https://doi.org/10.1007/11539117_147.
39. Yang Z, Tang K, Yao X. Large scale evolutionary optimization using cooperative coevolution. Inf Sci. 2008;178(15):2985–99. https://doi.org/10.1016/j.ins.2008.02.017.
40. Yang Z, Tang K, Yao X. Multilevel cooperative coevolution for large scale optimization. In: 2008 IEEE congress on evolutionary computation (IEEE World Congress on Computational Intelligence); 2008. p. 1663–70 . https://doi.org/10.1109/CEC.2008.4631014.
41. Omidvar MN, Li X, Yao X. Cooperative co-evolution with delta grouping for large scale non-separable function optimization. In: IEEE congress on evolutionary computation; 2010. p. 1–8 . https://doi.org/10.1109/CEC.2010.5585979.
42. Mahdavi S, Shiri ME, Rahnamayan S. Cooperative co-evolution with a new decomposition method for large-scale optimization. In: 2014 IEEE congress on evolutionary computation. New York: IEEE; 2014. p. 1285–92 . https://doi.org/10.1109/CEC.2014.6900327.
43. Omidvar MN, Mei Y, Li X. Effective decomposition of large-scale separable continuous functions for cooperative co-evolutionary algorithms. In: 2014 IEEE congress on evolutionary computation; 2014. p. 1305–12 . https://doi.org/10.1109/CEC.2014.6900420.
44. Omidvar MN, Li X, Mei Y, Yao X. Cooperative co-evolution with differential grouping for large scale optimization. IEEE Trans Evol Comput. 2014;18(3):378–93. https://doi.org/10.1109/TEVC.2013.2281543.
45. Sun Y, Kirley M, Halgamuge SK. Extended differential grouping for large scale global optimization with direct and indirect variable interactions. In: Proceedings of the 2015 annual conference on genetic and evolutionary computation. New York: ACM; 2015. p. 313–20. https://doi.org/10.1145/2739480.2754666.
46. Mei Y, Omidvar MN, Li X, Yao X. A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization. ACM Trans Math Softw. 2016. https://doi.org/10.1145/2791291.
47. Omidvar MN, Yang M, Mei Y, Li X, Yao X. DG2: a faster and more accurate differential grouping for large-scale black-box optimization. IEEE Trans Evol Comput. 2017;21(6):929–42. https://doi.org/10.1109/TEVC.2017.2694221.
48. Wu S, Zou Z, Fang W. A dynamic global differential grouping for large-scale black-box optimization. In: Tan Y, Shi Y, Tang Q, editors. Advances in swarm intelligence. Cham: Springer; 2018. p. 593–603. https://doi.org/10.1007/978-3-319-93815-8_56.
49. Sun Y, Omidvar MN, Kirley M, Li X. Adaptive threshold parameter estimation with recursive differential grouping for problem decomposition. In: Proceedings of the genetic and evolutionary computation conference. New York: ACM; 2018. p. 889–96. https://doi.org/10.1145/3205455.3205483.
50. Sun Y, Li X, Ernst A, Omidvar MN. Decomposition for large-scale optimization problems with overlapping components. In: 2019 IEEE congress on evolutionary computation (CEC); 2019. p. 326–33 . https://doi.org/10.1109/CEC.2019.8790204.
51. Mahdavi S, Shiri ME, Rahnamayan S. Metaheuristics in large-scale global continues optimization: a survey. Inf Sci. 2015;295:407–28. https://doi.org/10.1016/j.ins.2014.10.042.
52. Song A, Yang Q, Chen W, Zhang J. A random-based dynamic grouping strategy for large scale multi-objective optimization. In: 2016 IEEE congress on evolutionary computation; 2016. p. 468–75 . https://doi.org/10.1109/CEC.2016.7743831.
53. Mingming X, Jun Z, Kaiquan C, Xianbin C, Ke T. Cooperative co-evolution with weighted random grouping for large-scale crossing waypoints locating in air route network. In: Khoshgoftaar TM, Zhu X, editors. 23rd international conference on tools with artificial intelligence; 2011. p. 215–22 . https://doi.org/10.1109/ICTAI.2011.40.
54. Sun L, Lin L, Li H, Gen M. Cooperative co-evolution algorithm with an MRF-based decomposition strategy for stochastic flexible job shop scheduling. Mathematics. 2019;7(4):318. https://doi.org/10.3390/math7040318.
55. Ding W, Wang J. A novel approach to minimum attribute reduction based on quantum-inspired self-adaptive cooperative co-evolution. Knowl Based Syst. 2013;50:1–13. https://doi.org/10.1016/j.knosys.2013.03.008.

56. Yi L, Wu X, Li X, Cui X. A mean-field formulation for optimal multi-period mean-variance portfolio selection with an uncertain exit time. Oper Res Lett. 2014;42(8):489–94. https://doi.org/10.1016/j.orl.2014.08.007.
57. Jensen FV. Introduction to Bayesian networks. 1st ed. Berlin: Springer; 1996.
58. Cortes C, Vapnik V. Support-vector networks. Mach Learn. 1995;20(3):273–97. https://doi.org/10.1007/BF00994018.
59. Mucherino A, Papajorgji PJ, Pardalos PM. K-nearest neighbor classification. In: Data mining in agriculture. New York: Springer; 2009. p. 83–106. https://doi.org/10.1007/978-0-387-88615-2_4.
60. Xiaoliang Z, Hongcan Y, Jian W, Shangzhuo W. Research and application of the improved algorithm C4.5 on decision tree. In: 2009 international conference on test and measurement; New York: IEEE; 2009. vol. 2, p. 184–7 . https://doi.org/10.1109/ICTM.2009.5413078.
61. Breiman L. Random forests. Mach Learn. 2001;45(1):5–32. https://doi.org/10.1023/A:1010933404324.
62. Kurnaz FS, Hoffmann I, Filzmoser P. Robust and sparse estimation methods for high-dimensional linear and logistic regression. Chemom Intell Lab Syst. 2018;172:211–22. https://doi.org/10.1016/j.chemolab.2017.11.017.
63. Reddy GT, Reddy MPK, Lakshmanna K, Kaluri R, Rajput DS, Srivastava G, Baker T. Analysis of dimensionality reduction techniques on big data. IEEE Access. 2020;8:54776–88. https://doi.org/10.1109/ACCESS.2020.2980942.
64. van den Bergh F, Engelbrecht AP. A cooperative approach to particle swarm optimization. IEEE Trans Evol Comput. 2004;8(3):225–39. https://doi.org/10.1109/TEVC.2004.826069.
65. Trunfio GA, Topa P, Wąs J. A new algorithm for adapting the configuration of subcomponents in large-scale optimization with cooperative coevolution. Inf Sci. 2016;372:773–95. https://doi.org/10.1016/j.ins.2016.08.080.
66. Potter MA. The design and analysis of a computational model of cooperative coevolution. Ph.D. thesis, George Mason University, VA, United States; 1997.
67. Wiegand RP. An analysis of cooperative coevolutionary algorithms. Ph.D. thesis, George Mason University, VA, United States; 2003.
68. Storn R, Price K. Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim. 1997;11(4):341–59. https://doi.org/10.1023/A:1008202821328.
69. Bucci A, Pollack JB. On identifying global optima in cooperative coevolution. In: Proceedings of the 7th annual conference on genetic and evolutionary computation. New York: ACM; 2005. p. 539–44 . https://doi.org/10.1145/1068009.1068098.
70. Ambusaidi MA, He X, Nanda P, Tan Z. Building an intrusion detection system using a filter-based feature selection algorithm. IEEE Trans Comput. 2016;65(10):2986–98. https://doi.org/10.1109/TC.2016.2519914.
71. Ko I, Chambers D, Barrett E. Unsupervised learning with hierarchical feature selection for DDoS mitigation within the ISP domain. ETRI J. 2019;41(5):574–84. https://doi.org/10.4218/etrij.2019-0109.
72. Binbusayyis A, Vaiyapuri T. Identifying and benchmarking key features for cyber intrusion detection: an ensemble approach. IEEE Access. 2019;7:106495–513. https://doi.org/10.1109/ACCESS.2019.2929487.
73. Bagui S, Kalaimannan E, Bagui S, Nandi D, Pinto A. Using machine learning techniques to identify rare cyber-attacks on the UNSWNB15 dataset. Secur Priv. 2019;2(6):91. https://doi.org/10.1002/spy2.91.
74. Guyon I, Elisseeff A. An introduction to variable and feature selection. J Mach Learn Res. 2003;3:1157–82.
75. Xue B, Zhang M, Browne WN, Yao X. A survey on evolutionary computation approaches to feature selection. IEEE Trans Evol Comput. 2015;20(4):606–26. https://doi.org/10.1109/TEVC.2015.2504420.
76. Bommert A, Sun X, Bischl B, Rahnenführer J, Lang M. Benchmark for filter methods for feature selection in high-dimensional classification data. Comput Stat Data Anal. 2020;143:106839. https://doi.org/10.1016/j.csda.2019.106839.
77. Shukla AK, Tripathi D. Detecting biomarkers from microarray data using distributed correlation based gene selection. Genes Genom. 2020. https://doi.org/10.1007/s13258-020-00916-w.
78. Hancer E, Xue B, Zhang M. Differential evolution for filter feature selection based on information theory and feature ranking. Knowl Based Syst. 2018;140:103–19. https://doi.org/10.1016/j.knosys.2017.10.028.
79. Öziç MÜ, Özşen S. T-test feature ranking based 3D MR classification with VBM mask. In: 25th signal processing and communications applications conference. New York: IEEE; 2017. p. 1–4. https://doi.org/10.1109/SIU.2017.7960591.
80. John G, Kohavi R. Wrappers for feature subset selection. Artif Intell. 1997;97(1–2):273–324. https://doi.org/10.1016/S0004-3702(97)00043-X.
81. Wang A, An N, Chen G, Li L, Alterovitz G. Accelerating wrapper-based feature selection with k-nearest-neighbor. Knowl Based Syst. 2015;83:81–91. https://doi.org/10.1016/j.knosys.2015.03.009.
82. Bron EE, Smits M, Niessen WJ, Klein S. Feature selection based on the SVM weight vector for classification of dementia. IEEE J Biomed Health Inform. 2015;19(5):1617–26. https://doi.org/10.1109/JBHI.2015.2432832.
83. Maldonado S, López J. Dealing with high-dimensional class-imbalanced datasets: embedded feature selection for SVM classification. Appl Soft Comput. 2018;67:94–105. https://doi.org/10.1016/j.asoc.2018.02.051.
84. Tibshirani R. Regression shrinkage and selection via the lasso. J R Stat Soc Ser B. 1996;58(1):267–88.
85. Biau G, Cadre B, Rouvière L. Accelerated gradient boosting. Mach Learn. 2019;108(6):971–92. https://doi.org/10.1007/s10994-019-05787-1.
86. Tan CJ, Lim CP, Cheah Y-N. A multi-objective evolutionary algorithm-based ensemble optimizer for feature selection and classification with neural network models. Neurocomputing. 2014;125:217–28. https://doi.org/10.1016/j.neucom.2012.12.057.
87. Moslehi F, Haeri A. A novel hybrid wrapper-filter approach based on genetic algorithm, particle swarm optimization for feature subset selection. J Ambient Intell Humaniz Comput. 2020;11(3):1105–27. https://doi.org/10.1007/s12652-019-01364-5.
88. Nag K, Pal NR. A multiobjective genetic programming-based ensemble for simultaneous feature selection and classification. IEEE Trans Cybern. 2015;46(2):499–510. https://doi.org/10.1109/TCYB.2015.2404806.
89. Soufan O, Kleftogiannis D, Kalnis P, Bajic VB. DWFS: a wrapper feature selection tool based on a parallel genetic algorithm. PLoS ONE. 2015. https://doi.org/10.1371/journal.pone.0117988.
90. Song XF, Zhang Y, Guo YN, Sun XY, Wang YL. Variable-size cooperative coevolutionary particle swarm optimization for feature selection on high-dimensional data. IEEE Trans Evol Comput. 2020. https://doi.org/10.1109/TEVC.2020.2968743.

91. Kashef S, Nezamabadi-pour H. An advanced ACO algorithm for feature subset selection. Neurocomputing. 2015;147:271–9. https://doi.org/10.1016/j.neucom.2014.06.067.
92. Shukla AK. Feature selection inspired by human intelligence for improving classification accuracy of cancer types. Comput Intell. 2020. https://doi.org/10.1111/coin.12341.
93. Shukla AK, Singh P, Vardhan M. A new hybrid feature subset selection framework based on binary genetic algorithm and information theory. Int J Comput Intell Appl. 2019;18(03):1950020. https://doi.org/10.1142/S1469026819500202.
94. Shukla AK, Singh P, Vardhan M. Gene selection for cancer types classification using novel hybrid metaheuristics approach. Swarm Evol Comput. 2020;54:100661. https://doi.org/10.1016/j.swevo.2020.100661.
95. Zorarpacı E, Özel SA. A hybrid approach of differential evolution and artificial bee colony for feature selection. Expert Syst Appl. 2016;62:91–103. https://doi.org/10.1016/j.eswa.2016.06.004.
96. Zhang M, Ma J, Gong M, Li H, Liu J. Memetic algorithm based feature selection for hyperspectral images classification. In: 2017 IEEE congress on evolutionary computation. New York: IEEE; 2017. p. 495–502 . https://doi.org/10.1109/CEC.2017.7969352.
97. Han M, Ren W. Global mutual information-based feature selection approach using single-objective and multi-objective optimization. Neurocomputing. 2015;168:47–54. https://doi.org/10.1016/j.neucom.2015.06.016.
98. Hamdani TM, Won J-M, Alimi AM, Karray F. Multi-objective feature selection with NSGA II. In: International conference on adaptive and natural computing algorithms. Berlin: Springer; 2007. p. 240–7. https://doi.org/10.1007/978-3-540-71618-1_27.
99. Yuan Y, Xu H, Wang B. An improved NSGA-III procedure for evolutionary many-objective optimization. In: Proceedings of the 2014 annual conference on genetic and evolutionary computation; 2014. p. 661–8 . https://doi.org/10.1145/2576768.2598342.

Publisher's Note