International Journal of Grid and High Performance Computing
Volume 15 • Issue 1

A Parallel Hybrid Feature Selection Approach Based on Multi-Correlation and Evolutionary Multitasking

Mohamed Amine Azaiz, LabRI Laboratory, École Supérieure en Informatique de Sidi Bel Abbes, Algeria*
https://orcid.org/0000-0003-3792-0356

Djamel Amar Bensaber, LabRI Laboratory, École Supérieure en Informatique de Sidi Bel Abbes, Algeria

ABSTRACT

Particle swarm optimization (PSO) has been successfully applied to feature selection (FS) due to its efficiency and ease of implementation. Like most evolutionary algorithms, they still suffer from a high computational burden and poor generalization ability. Multifactorial optimization (MFO), as an effective evolutionary multitasking paradigm, has been widely used for solving complex problems through implicit knowledge transfer between related tasks. Based on MFO, this study proposes a PSO-based FS method to solve high-dimensional classification via information sharing between two related tasks generated from a dataset using two different measures of correlation. To be specific, two subsets of relevant features are generated using symmetric uncertainty measure and Pearson correlation coefficient, then each subset is assigned to one task. To improve runtime, the authors proposed a parallel fitness evaluation of particles under Apache Spark. The results show that the proposed FS method can achieve higher classification accuracy with a smaller feature subset in a reasonable time.

KEYWORDS

Big Data Analysis, Evolutionary Multitasking, Feature Selection, High-Dimensional Data Classification, Parallel Binary PSO, Parallel Computation

INTRODUCTION

Big data has contributed to the complexity of analysis algorithms by increasing the dimension of data. Feature selection is a pre-processing step in the classification process, aimed at reducing the dimension of data to improve learning performance (Rong et al., 2019). Researchers have presented different approaches to selecting features (Jovi´c et al., 2015), which can be grouped into Filter-based, Wrapper-based and Hybrid-based, and Embedded-based approaches. Filter Methods uses statistical tests to select features based on their individual contribution to the prediction task. Examples of filter methods include chi-squared tests, correlation-based feature selection, and mutual information-based feature selection. Many filter methods are based on some mathematical statistics concepts to measure the degree of correlation (or statistical independence) between different features, and also

DOI: 10.4018/IJGHPC.320475 *Corresponding Author

between features and target class. X is a redundant feature if it has a strong correlation with another feature Y (Yu et al., 2003), In this case, we can dispense with one of the two features, since the information generated by X can be inferred from Y. An irrelevant feature is a feature that has weak or no correlation with the target class (Song et al., 2013), We can also dispense with this type of feature, because it negatively affects predictive accuracy. Correlation based filter methods are characterized by the speed of execution, but good results are not always guaranteed due to the insufficiency of a unified and comprehensive definition of statistical correlation. For example, two variables which are not linearly correlated are not necessarily independent, it is possible that they are non-linearly correlated. This is why we propose in our approach the use of multi-correlation to avoid losing some features that contain important information. Wrapper-based approaches search through the feature space by using a learning algorithm to evaluate selected features. These methods are characterized by the quality of their results in most cases, but they are challenged by their complexity and execution time. Hybrid methods use filter and wrappers approaches to combine fast execution with quality of results. Embedded methods integrate feature selection into the training process of machine learning algorithms. Examples of embedded methods include Lasso Regression and Random Forest feature importance. Hybrid Methods combine aspects of Filter, Wrapper, and Embedded methods to create a new feature selection approach. Examples of hybrid methods include Feature Selection using the combination of Recursive Feature Elimination and SVM.

Feature selection is the process of finding a subset of features in a large space, which is defined as the combinatorial optimization problem. In such cases, the use of evolutionary algorithms is among the most effective solutions. Particle swarm optimization (PSO) (Kennedy et al., 1995), as a population-based search algorithm has been widely applied for solving feature selection problems. This is because PSO has the advantages of being easy to implement and has strong global searchability. The Binary PSO algorithm is a variant of the PSO algorithm (Kennedy et al., 1997) to solve discrete optimization problems. Despite the achievements obtained by the evolutionary algorithms, they require massive computational resources to guarantee the convergence performance compared to other optimization methods. Multitasking evolutionary optimization (Gupta et al., 2017) is a recent approach based on sharing and disseminating knowledge between different and related problems (tasks) and helping solve problems during the research process. Multifactorial optimization (MFO) is an evolutionary multitasking paradigm introduced by Gupta et al. in 2015 (Gupta et al., 2017). Multifactorial PSO (MFPSO) (Feng et al., 2017) was proposed under the MFO paradigm, which effectively shares knowledge across related tasks through the operators of assortative mating and vertical cultural transmission during the research process.

Since there are multiple definitions to measure the degree of correlation between two variables, we suggested in our approach the use of two different measures of correlation: the first is a non-linear measure using Symmetric Uncertainty, and the second is linear measure using the coefficient of Pearson (Chandra et al., 2012), therefore, we have two sets of relevant features. By exploiting Evolutionary multitasking, we will create two related tasks: The first is to search the appropriate features in the relevant feature subset generated by symmetric uncertainty measure, and the second is to search the appropriate features in the relevant feature subset generated by Pearson coefficient. Apache Spark (Shaikh et al., 2019) is one of the best open-source unified analytics engines for large scale data processing based on various big data technologies such as the MapReduce framework. In the context of big data, we are dealing with data that contains thousands of features, which makes the complexity of execution very high, and thus affecting the performance of the algorithm. In our approach, we used Apache Spark to calculate the correlation values between each feature and the target class in parallel. In a PSO-based FS approach, the sequential fitness evaluation of all particles is very time consuming, so we have proposed a parallel evaluation under Apache Spark. In this work, we propose a feature selection approach in the context of big data to solve the classification problem of high-dimensional data. Specifically, the main contributions of this work are given as follows:

- We proposed a hybrid method for selecting features for high-dimensional data, based on the evolutionary multitask optimization using Binary PSO under MFO paradigm (MF-BPSO). We called our proposed algorithm: PHFS-EMT (Parallel Hybrid Feature Selection- Evolutionary Multitasking).
- We Combine two different measures of correlation to find relevant features, to avoid losing some features that contain important information.
- We Proposed a parallel fitness evaluation of the set of particles under Apache.

The rest of this paper is organized as follows: in section 2, we review related works. Then, we explain some of the basic concepts used in our approach, such as correlation measurement, and the BPSO algorithm in section 3. In section 4, we present our proposed approach. In section 5, the obtained results are analyzed and discussed. Lastly, in section 6, we conclude the paper and provide some future perspectives.

RELATED WORKS

In the past few years, a large number of PSO-based FS approaches have been developed for high-dimensional classification problems (Xue et al., 2016), (Xue et al., 2013). Filter and Wrapper methods have been developed based on various strategies to improve the performance of PSO-based FS approaches on high-dimensional classification problems. In Filter methods, several metrics have been introduced for selecting subsets of features, such as entropy (Dai et al., 2013) and mutual information. In Wrapper methods, subsets of features are evaluated by the results of a learning algorithm, such as Naïve Bayes and k-nearest neighbor (Chen et al., 2019). The hybrid approach (Tran et al., 2016) is a type of FS method that combines filter-based and wrapper-based approaches. As their performance is better than using filter-based and Wrapper methods alone, hybrid FS methods have attracted wide interest from researchers.

Reducing the search space by removing redundant and irrelevant features is a useful way to improve PSO's search performance. A new hybrid based on PSO using the Gaussian distribution has been proposed in (Lane et al., 2014) to address FS problems. This method allocates features in the corresponding group according to the statistical clustering information. Then, a certain number of features were selected from each group during the evolution process. Experimental results showed that this method can choose a smaller feature set to get better or similar performance from original feature set.

In (Tran et al., 2019), a variable-length representation based on the relevance of features was developed. In this method, a filter approach was applied to initial particles with different lengths using symmetric uncertainty measure, then, the k-nearest neighbor classifier evaluates the selected feature subset. Moreover, a length changing mechanism was also proposed to escape from local optima during the search process. The experimental results showed that this FS approach can achieve a smaller feature subset with best accuracy than the compared fixed length FS methods. But the method with the length varying mechanism, may cause the search to fall into a local optimum due to the high speed of feature deletion during the selection process. In (Tran et al., 2019), an adaptive multisubswarm PSO for FS on the high-dimensional classification problems was proposed. This method is based on dividing the search space into smaller subspaces according to the importance of the features. Subswarms change based on their performance automatically during the search process. The results showed a high effectiveness in terms of classification accuracy and number of features selected. However, the insufficiency of information interchange between subswarms may reduce search efficiency during the FS process.

Instead of developing filter and wrapper approaches in two different phases in the search process, some hybrid FS methods combined filter and wrapper approaches in a one phase to make better the

performance of PSO for the high dimensional classification problems. In (Lui et al, 2013), a fast hybrid FS method based on PSO and mutual information was proposed. The results showed that this method can reach greater performance in terms of classification accuracy in most cases. In (Banka et al., 2015), mutual information was used to score the degree of redundancy and relevance of a feature, and then this score was used to lead the updating of velocity. The results showed that this method can reach greater performance in terms of classification accuracy in most cases.

(Chen et al., 2020), proposed a method to solve high-dimensional classification via information sharing between two related tasks generated from a dataset. So, two related tasks about the target class are established by evaluating the importance of features. A new crossover operator, called assortative mating, is applied to share information between these two related tasks. In addition, two mechanisms, which are variable-range strategy and subset updating mechanism, are also developed to reduce the search space and maintain the diversity of the population, respectively. Although a many hybrid PSO based FS approaches have been proposed to increase PSO's search performance on the high-dimensional classification problems, not many studies solve the limitations of trapping into local optima and high computational time simultaneously. PSO for FS in high dimensional data still needs more studies.

(Arie et al., 2021) Proposed an approach to select best features subset using Wrapper method for diabetes prediction dataset which has been transformed to numeric dataset previously. Backward and forward selection are used in wrapper method, that's combine with random forest and cross validation. Random forest is decision tree improvement, which is group of trees that can produce difference or same result at each tree. The most results are made as final result. The final result from feature selection with wrapper method can make higher accuracy than without feature selection for numeric dataset and the number of features can be reduced.

In (Oğuzhan et al., 2021), An evolutionary based optimization algorithm utilizing self-organizing map was accordingly modified to create a new feature selection algorithm for the classification of hyperspectral images.

Normalized mutual information (NMI) and Jaya Algorithm are combined in (Kiran et al., 2022) to develop a hybrid feature selection method. The Normalized Mutual Information Jaya Algorithm (NMIJA) selection approach significantly decreases data dimension and minimizes classification redundancies. The reduced dataset achieves the highest classification accuracy when compared to typical feature selection algorithms.

(Wenyu et al., 2022) proposes a classification approach of brain disease based on an improved sparrow search algorithm (ISSA). Specifically, a group of image features is first extracted to compose a pool of features for selection from five kinds of brain hemorrhage image datasets and one of the brain tumor image datasets. Secondly, an objective function is proposed to jointly reduce the number of the selected features and improve the classification accuracy. The proposed ISSA is utilized to solve the objective function. ISSA introduces three improvement mechanisms: the tent chaotic initialization, a novel local search strategy, and adaptive crossover operation to enhance the performance of conventional SSA. However, a binary operator is used to solve discrete feature selection problems.

BACKGROUND

In this section, we will review some of the concepts and definitions that we used in our study.

Pearson Correlation Coefficient

Pearson's coefficient (Chandra et al., 2012) is an index reflecting a linear relationship between two variables. The correlation coefficient varies between -1 and +1, 0 reflecting a zero relationship between the two variables, a negative value (negative correlation) meaning that when one of the variables increases, the other decreases; while a positive value (positive correlation) indicates that

the two variables vary together in the same direction. Here is the formula to calculate the Pearson correlation coefficient:

$$\mathbf{r} = \frac{\sum_{\mathbf{i}=1}^{\mathbf{n}} \left(\mathbf{X}_{\mathbf{i}} - \bar{\mathbf{X}}\right)\left(\mathbf{Y}_{\mathbf{i}} - \bar{\mathbf{Y}}\right)}{\sqrt{\sum_{\mathbf{i}=1}^{\mathbf{n}} \left(\mathbf{X}_{\mathbf{i}} - \bar{\mathbf{X}}\right)^{2}} \sqrt{\sum_{\mathbf{i}=1}^{\mathbf{n}} \left(\mathbf{Y}_{\mathbf{i}} - \bar{\mathbf{Y}}\right)^{2}}} \tag{1}$$

Where, $\bar{X}$ is the mean of X, Y is the mean of Y.

The value of r obtained is an estimate of the correlation between two continuous variables in the population. Therefore, its value will fluctuate from sample to sample. We therefore want to know if, in the population, these two variables are really correlated or not.

Symmetric Uncertainty Measure

To understand Symmetric Uncertainty, the following concepts must be defined: Entropy, Mutual Information. Entropy, or Shannon's entropy (Song et al., 2013), is an essential concept in information theory, which intuitively corresponds with the quantity of information contained or delivered by a source of information. Mathematically, the entropy of a discrete variable X is defined by the following equation:

$$\mathbf{H}\left(\mathbf{X}\right) = -\sum_{\mathbf{x}\in\mathbf{X}} \mathbf{p}\left(\mathbf{x}\right) \mathbf{log}_{2}\left(\mathbf{x}\right) \tag{2}$$

Where, $p\left(x\right)$ is the probability of x.

Conditional entropy noted by $H(X \mid Y)$, measures the amount of information needed to describe the outcome of a random variable Y given that the value of another random variable Y is known. Conditional entropy is a symmetric measure, which is defined by the following formula:

$$\mathbf{H}\left(\mathbf{X} \mid \mathbf{Y}\right) = -\sum_{\mathbf{y}\in\mathbf{Y}} \mathbf{p}\left(\mathbf{y}\right) \sum_{\mathbf{x}\in\mathbf{X}} \mathbf{p}\left(\mathbf{x}|\mathbf{y}\right) \mathbf{log}_{2} \mathbf{p}\left(\mathbf{x}|\mathbf{y}\right) \tag{3}$$

Where, Where, $p(x|y)$ is the posterior probability of X.

Mutual information (Song et al., 2013) measures the amount of statistical dependence between two random variables. Information gain is the amount of information gained about a random variable X observing another random variable Y, so it is defined by following equation:

$$\mathbf{Gain}\left(\mathbf{X}|\mathbf{Y}\right) = \mathbf{H}\left(\mathbf{X}\right) - \mathbf{H}\left(\mathbf{X}|\mathbf{Y}\right) = \mathbf{H}\left(\mathbf{Y}\right) - \mathbf{H}\left(\mathbf{Y}|\mathbf{X}\right) \tag{4}$$

Where, $H(X)$ is the entropy of a random variable X and $H(X|Y)$ is the conditional entropy. Symmetric uncertainty normalizes the value of information gain in range [0,1] as follows:

$$\mathbf{SU}\left(\mathbf{X}, \mathbf{Y}\right) = \frac{2 * \mathbf{Gain}\left(\mathbf{X}|\mathbf{Y}\right)}{\mathbf{H}\left(\mathbf{X}\right) + \mathbf{H}\left(\mathbf{Y}\right)} \tag{5}$$

Particle Swarm Optimization Algorithm (PSO)

This algorithm (Kennedy et al., 1995) is inspired by the origin of the living world. It is based on a model developed by Craig Reynolds in the late 1980s, which simulates the movement of a group of birds. Another source of inspiration, claimed by the authors, James Kennedy and Russel Eberhart, is socio-psychology in 1995. This optimization method is based on the collaboration of individuals called" Particles" between them. Thus, thanks to very simple displacement rules (in the solution space), the particles can gradually converge towards a global minimum. However, this metaheuristic seems to work better for spaces in continuous variables. At the beginning of the algorithm, each particle is therefore positioned (randomly or not) in the search space for the problem. Each iteration makes the particles move according to 3 components: Its current velocity, his best personal solution ($pbest$), and the best solution obtained in its neighborhood ($gbest$). In a search space with a d dimension, we can define the position of the particle i, at the moment t as follows $:X_i\left(t\right)=\left(X_{i1},X_{i2},...,X_{iD}\right)$, and its velocity by: $V_i\left(t\right)=\left(V_{i1},V_{i2},...,V_{iD}\right)$. Each particle adjusts its velocity and position according to the velocity and position of the particle itself, the perceptive and the social part, as shown in the following equations:

$$\mathbf{v}_{\mathbf{id}}\left(\mathbf{t}+1\right)=\mathbf{w}.\mathbf{v}_{\mathbf{id}}\left(\mathbf{t}\right)+\mathbf{c}_1.\mathbf{r}_1.\left(\mathbf{pbest}_{\mathbf{id}}\left(\mathbf{t}\right)-\mathbf{x}_{\mathbf{id}}\left(\mathbf{t}\right)\right)+\mathbf{c}_2.\mathbf{r}_2.\left(\mathbf{gbest}_{\mathbf{d}}\left(\mathbf{t}\right)-\mathbf{x}_{\mathbf{id}}\left(\mathbf{t}\right)\right) \tag{6}$$

$$\mathbf{x}_{\mathbf{id}}\left(\mathbf{t}+1\right)=\mathbf{x}_{\mathbf{id}}\left(\mathbf{t}\right)+\mathbf{v}_{\mathbf{id}}\left(\mathbf{t}+1\right) \tag{7}$$

Where c_1 is the cognitive learning factor and c_2 is the social learning factor; r_1 and r_2 are random numbers uniformly distributed in [0,1]; w is the weight of inertia. The fitness function is a very important concept regarding the PSO algorithm. This function evaluates the fitness values of each particle in order to guide the search for the best global solution.

Binary Particle Swarm Optimization (BPSO)

PSO has been successfully applied to continuous non-linear functions and most applications have focused on solving continuous optimization. To solve discrete (combinatorial) optimization problems, Kennedy and Aberhart proposed a separate PSO (BPSO) (Kennedy et al., 1997), with each particle containing a suit of 0 and 1. The velocity value is limited in the interval [0,1] using the sigmoid function:

$$\mathbf{s}\left(\mathbf{v}_{\mathbf{id}}\right)=\frac{1}{1+\mathbf{e}^{-\mathbf{v}_{\mathbf{id}}}} \tag{8}$$

Where $s\left(v_{id}\right)$ indicates the probability that the x_{id} bit will take the value 1.

The particle, then, changes its bit value randomly by:

$$\mathbf{x}_{\mathbf{id}}=\begin{cases}1 if\, rand\le s\left(\mathbf{v}_{\mathbf{id}}\right)\\ 0 otherwise\end{cases} \tag{9}$$

Where $rand$ is a random number chosen from a uniform distribution in the interval $\left[0,1\right]$. To avoid the approach of $s\left(v_{id}\right)$ from 1 or 0, the velocity v_{id} must be limited between the V_{min} and V_{max} constants, so that V_{min} represents the minimum velocity and V_{max} the maximum velocity, such that $v_{id} \in [V_{max}, V_{max}]$.

Evolutionary Multitasking

Evolutionary multitasking (Gupta et al., 2017) is a recent approach that uses evolutionary algorithms to find solutions to several tasks in real time, by exchanging knowledge between these tasks. The importance of evolutionary multitasking is that addressing one task contributes to finding solutions to other tasks, because there is often common knowledge and problems between these tasks. For example, the multitasking framework contains K tasks. Task k is denoted as T_K and its search space is X_K with the fitness function f_k. The multi-task framework can be described as:

$$\left(\mathbf{x}_1^*, \mathbf{x}_2^*, \ldots, \mathbf{x}_\mathbf{k}^*\right) = \mathbf{argminf}_1, \mathbf{argminf}_2, \ldots, \mathbf{argminf}_\mathbf{k} \tag{10}$$

Where x_k^* denotes the optimal global solution of T_K.

Multifactorial Evolution (MFO) (Feng et al., 2017) is a paradigm for implementing evolutionary multitasking. In MFO, a unified random key scheme is used to encode the appropriate search space of specific tasks into a unified representation space. This means that each individual in the population can be decoded into a task specific solution for every task. In MFO there are some important definitions that you should know, and they are as follows:

- **Factorial Cost:** The factorial cost on task T_K of individual p_i is the fitness value of solution p_i.
- **Factorial Rank:** Factorial rank of p_i in the task T_K is the index of T_K in a list of individuals arranged in descending order according to each individual's fitness value, which is noted as r_i^k.
- **Skill Factor:** The skill factor i of p_i is the index of the task in which the individual p_i gets the best fitness value.
- **Scalar Fitness:** The scalar fitness of pi is given by $\varphi_i = \dfrac{1}{\min\left(r_i^k\right)}$. The skill factor is applied to allocate the task for a specific individual during the search process, so it's very important.

Multifactorial Particle Swarm Optimization (MFPSO)

MFPSO (Feng et al., 2017) is the implementation of the PSO algorithm in a multitasking evolutionary context under MFO paradigm. In the initialization part, all particles are evaluated on all tasks. Then in each iteration and by using vertical cultural transmission, each particle is directed to one task depending on the value of the skill factor. For transferring information between individuals with different tasks, MFPSO use Assortative mating and vertical cultural transmission. Assortative mating is a procedure that compares the value of a variable called rmp with a random variable belonging to the field $\left[0,1\right]$, so that if the value of rmp is greater, the velocity of the particle changes using equation (6) instead of equation (11):

$$\mathbf{v}_{\mathbf{id}}\left(\mathbf{t}+1\right) = \mathbf{w}.\mathbf{v}_{\mathbf{id}}\left(\mathbf{t}\right) + \mathbf{c}_1.\mathbf{r}_1.\left(\mathbf{pbest}_{\mathbf{id}}\left(\mathbf{t}\right) - \mathbf{x}_{\mathbf{id}}\left(\mathbf{t}\right)\right) + \mathbf{c}_2.\mathbf{r}_2.\left(\mathbf{gbest}_{\mathbf{id}}\left(\mathbf{t}\right) - \mathbf{x}_{\mathbf{d}}\left(\mathbf{t}\right)\right) + \mathbf{c}_3.\mathbf{r}_3.\left(\mathbf{gbest}_{\mathbf{id}}^{'}\left(\mathbf{t}\right) - \mathbf{x}_{\mathbf{d}}\left(\mathbf{t}\right)\right) \tag{11}$$

Where c_3 is the cognitive learning factor and r_3 is a random number uniformly distributed in $[0,1]$. $gbest'_{id}$ is the global best position of another task. The importance of this equation is the use of information that comes from other tasks.

The procedure of MFPSO is described in (Chen et al., 2020). For more details about MFPSO see (Feng et al., 2017).

THE PROPOSED APPROACH

In this section, we propose a hybrid feature selection method based on multitasking evolutionary and multi-correlation for high-dimensional classification problems. Specifically, two relevant sets of features are generated using symmetric uncertainty and Pearson's linear coefficient. Each set is assigned to one task and then by using evolutionary multitasking the appropriate feature set is selected.

Step 1: Generating Two Related Tasks

When using evolutionary multitasking for handling FS on the high-dimensional classification problems, the relevance of constructed tasks is an important factor affecting the classification performance. In a multitasking method, the tasks should satisfy a certain degree of commonality or complementarity in terms of the optimal solution or function landscape.

In our study, two tasks with different search spaces based on different relevant feature sets are developed. To create the first set, the correlation values between each feature and the target class are calculated using Symmetric Uncertainty. Then, the most relevant features are selected, which represent the first set of relevant features. With the same way, the second set of relevant features is created, but using the Pearson's Linear Correlation measure. Algorithm 1 further explains how to create the two sets of relevant features.

Algorithm 1: Generating two set of relevant features

Input: $D(F_1, F_2, ..., F_n, C)$: dataset, θ_1 and θ_2, Threshold values; N, number of features

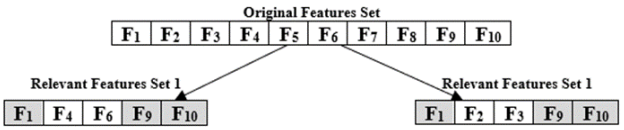
Output: S, First relevant feature set; R: Second relevant feature set

```
for i = 1 to N do
    Calculate su(F_i, N) using equation (5)
    Calculate r(F_i, N) using equation (1)
    if su(F_i, N) >= θ_1 then
        append F_i in S
    if r(F_i, N) >= θ_2 then
        append F_i in R
```

And therefore task 1 is to select features from a first subset of relevant features (S), and Task 2 is to select features from a second subset of relevant features (R).

Figure 1 illustrate an example of creating two different sets of relevant features with some common features. F_1, F_4, F_6, F_9, and F_{10} is the relevant features set 1 constructed using the Symmetric uncertainty measure, while F_1, F_2, F_3, F_9, F_{10} is the relevant features set constructed using the Pearson linear correlation coefficient. Since task 1 and task 2 have some features in common (We will see in the experiments section the different common features between the two relevant subsets for different datasets) and they also have the same tasks to predict the same set of class labels, these two FS tasks have a certain degree in common in terms of best solution. According to the conditions of evolutionary

Figure 1. Example is used to illustrate reducing the dimension of the original set and creating two related sets of features with some common features



multitasking mentioned above, relevant features set 1 and relevant features set 2 are related tasks in multitasking. During the FS process, Task 1 focuses on searching for important features in relevant features set 1, while task 2 ensures that important features are found in relevant features set 2. These two tasks can help each other by sharing information between different search spaces.

Step 2: Applying Evolutionary Multitasking for FS using BPSO

1. **Knowledge Transfer:** One of the advantages of a multitasking evolutionary is the transfer and sharing of knowledge between tasks. In this work, we have adopted a multipopulational framework for the sharing of knowledge between Task 1 and Task 2. So, at the beginning of the algorithm, the fitness value of each particle of the swarm is evaluated in both tasks. Then during all iterations, each particle is assigned to a task based on the skill factor value, this means, if a particle whose skill factor is 1 then the particle is assigned to process task 1; Otherwise, it is used to process task 2. Vertical cultural transmission is a factor which can change the value of skill factor to achieve transfer solutions between Task 1 and Task 2. The method of knowledge transfer plays an important role in knowledge transfer, which greatly influences the quality of the final selected subset. Knowledge is transferred between the two tasks via a crossover called Assortative Mating. During the evolution process, the random mating probability (rmp) is defined to control when knowledge is transferred from another task. In each generation, if the random number of $rand$ is less than or equal to rmp , (11) is adopted to update the velocity of particle; Otherwise, (6) is applied to update the velocity of particle. See Algorithm 2 for more explanation.

Algorithm 2: Assortative Mating

Input: c_1, c_2, c_3, three acceleration constants; rmp,random mating probability

Output: The velocity of each particle

```
Generate a random number rand between 0 and 1.;
if rand < rmp then
    Update the velocity of particle using (11)
else
    Update the velocity of particle using (6)
```

2. **Fitness Function:** The fitness function is a very important concept in the BPSO algorithm, it plays an important role for guiding the search process to find the global optimal solution. Therefore, the particle represents a relevant set of features, so that each particle moves in a space of dimension equal to number of relevant features. The coordinates of each particle's position are a series of values of 0 or 1 (These values are obtained by equations (6), (8), (9), (11)), where a value of 1 means that the feature is selected, and a value of 0 means that the feature is not, as shown in Figure 2. At the start of the algorithm, the values of the coordinates of each particle

Figure 2. Particle location in a m-dimensional space, representing selected features by the value 1 and canceled features with the value 0

F_1	F_2	....	F_i	...	F_r
0	1	...	1	...	0

are randomly determined, and then each particle changes its coordinates by using equations (6), (8), (9), and (11) according to each case .

So, we have proposed equation (12) which represents our fitness function:

$$\mathbf{f} = \mathbf{accuracy} - \pm\cdot\mathbf{pcFS} \tag{12}$$

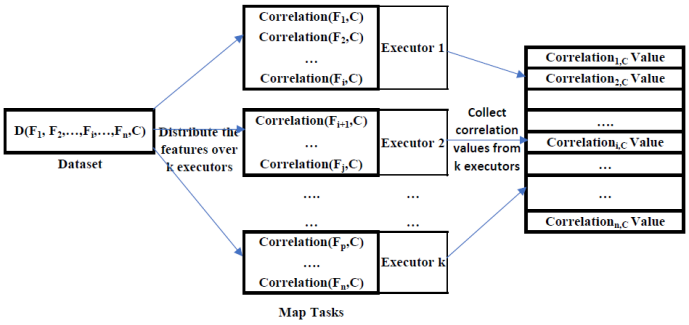
So that accuracy represents the classification accuracy of the selected feature subset (after removing all features that represent coordinates 0) in a specific task. α is a number in the field [0,1], and $pcFS$ is a percentage of the selected features as shown in the equation (13):

$$\mathbf{pcFS} = \frac{\mathbf{nbFS}}{\mathbf{N}}.100 \tag{13}$$

Where $nbFS$ is the number of selected features, and N is the total number of all features. The algorithm tries each time to find the largest possible value for f , which means finding a larger value for accuracy and a smaller value for the number of features selected ($pcFS$). Which means that our fitness function is trying to improve two parameters at the same time: increasing the accuracy and reducing the number of features, which is the purpose of the feature selection process. Although the algorithm improves both parameters, the algorithm prioritizes improving accuracy further, therefore, we make the parameter α to control the effect of $pcFs$ value on accuracy. If the $pcFS$ value negatively affects the accuracy, we can set α a smaller value. We can consider α as a coefficient increasing and decreasing in the interval $\left[0,1\right]$ according to each case. In each iteration, the algorithm tries to maximize the value of $f,$ so, the best personal solution ($pbest$) for the particle i , is the position whose coordinates give us the greatest value for f . The global best solution ($gBest$) is the best of all pBest which gives us the largest value of f , which is our ultimate solution, and that's for every task.

3. **Final Selected Features:** It is known that the ultimate objective of this study is to obtain a feature set that achieves lower dimension with high classification accuracy in the context of high dimensional classification problems. However, two subsets of features will be produced when evolutionary multitasking is applied, which are from task 1 and task 2, respectively. We will choose the selected features from the task that give us the highest accuracy.
4. **Parallel Processing:** Since we are in the context of big data, we are processing data with thousands of features, which consumes a lot of time which negatively affects the overall performance of our proposed algorithm. Apache Spark is a big data analysis engine used by many researchers and developers, and it works under several frameworks, the most famous of which is MapReduce. To speed up the algorithm execution, we implemented our algorithm under Spark in both phases. In the first stage, the features are randomly distributed into groups, and then each group is placed in a Spark executor. In each group, the correlation values between each feature of this group and class target are calculated and then Apache Spark executes all the executors at the same time, as shown in figure 3. The number of groups is equal to the number of available Spark executors.

Figure 3. Parallel calculation of correlation under Apache Spark



In the second phase, which is most important, and using the same principle. We randomly distribute the particles into groups, and then each group is placed in a Spark executor. Within each group, the fitness value of each particle is evaluated, then Apache Spark runs all the executors in parallel as shown in figure 4. The number of groups is equal to the number of available Spark executors. The only part of the PSO algorithm that we paralleled with is the part related to calculating the fitness value of particles, because in PSO-Based FS, we often use a classifier inside the fitness function, which takes a lot of time.

Figure 5 shows the general view of our approach. The proposed algorithm details are shown in Algorithm (2).

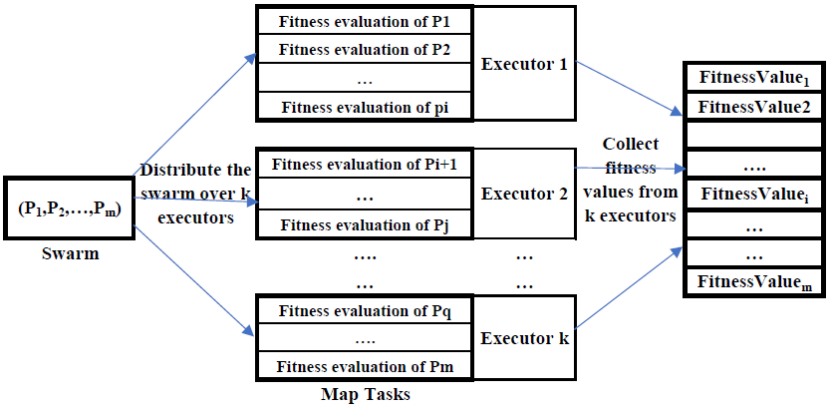
RESULTS AND DISCUSSION

This section describes the details of the experimental setup, including the examined data sets, comparison methods, and parameter settings.

Environment

To test the effectiveness of our approach, we implemented our algorithm using Java and Scala under a computer with the following characteristics: Processor: Intel® Core™ i7-8700 @ 3.20GHZ x 6, RAM: 32.00Go, Operating system: Linux Mint 20.1 x64. We also used Apache Spark for parallel

Figure 4. The proposed parallel fitness evaluation of BPSO under Apache Spark



processing under the same machine. The number of executors is determined by Apache Spark, which in our case is 11. We used the classifiers KNN and Naïve Bayes from WEKA API (Hall et al, 1994).

Algorithm 3: Parallel Hybrid Feature Selection

Input: $D(F_1, F_2, ..., F_n, C)$: dataset, θ_1, θ_2, α : Float //θ_1 and θ_2 in]0,1], α in [0,1]
Output: Best features subset

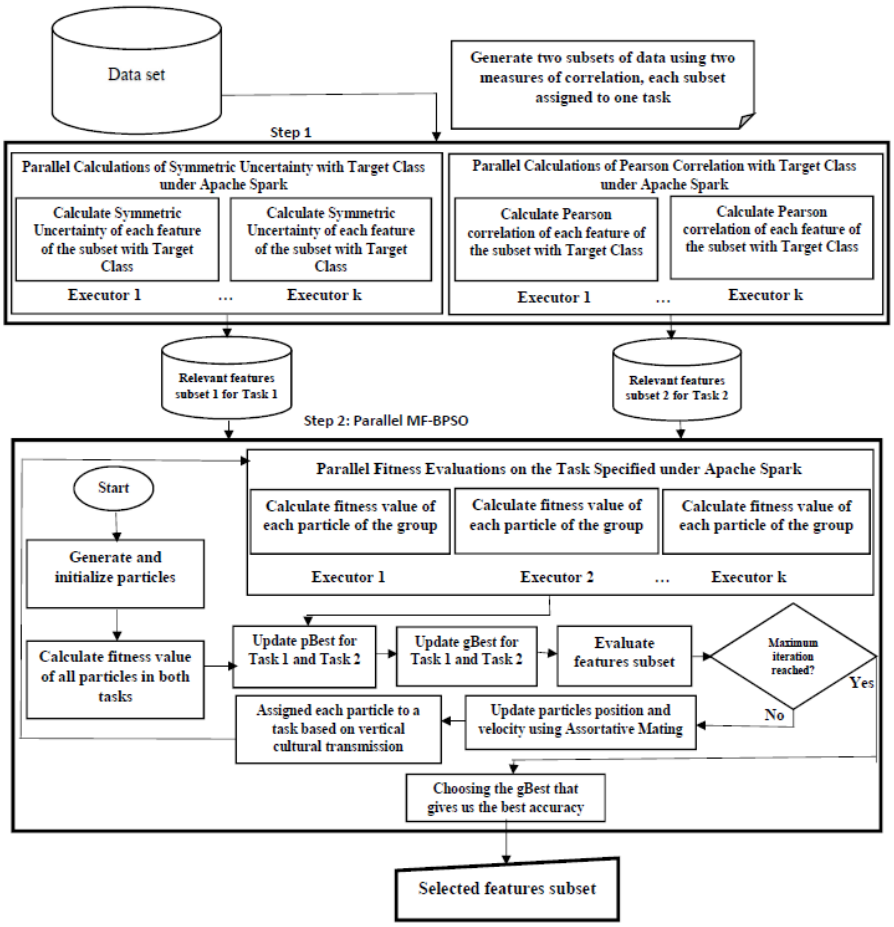
Function `Main`:
  /*Step 1: Reducing Search Space and Generating two Related Tasks*/
  Start parallel calculation of correlation values using equations (1), (5) under Apache Spark
  foreach $F_i \in$ D *append* F_i to S if $su(F_i, C) >= \theta_1$
  foreach $F_i \in$ D *append* F_i to R if $r(F_i, C) >= \theta_2$
  Assigned S to Task1, and R to Task2
  FS=ParallelMFBPSO() /* Step 2 find a best features subset using Parallel MF-BPSO */
  return FS
Function `ParallelMFBPSO()`:
  Input: $C_1, C_2, w, V_{min}, V_{max}$ /*The number of particles in each group, as well as the number of groups is determined by Spark.*/
  Start initialize population of particles randomly with a binary random positions and velocities in a space with a r dimension /* r is the number of relevant features */
  Evaluate all particles in each Task
  while *maxIteration* **do**
    Calculate SkillFactor for each particle
    Calculate ScalarFitness for each particle
    Assigned each particle to a task based on vertical cultural transmission
    Distribute each group of particles over k spark Executors
    Start parallel execution of k executors
    In each executor for each particle fitnessValue = calculateFitnessValue()
    Collect fitness values
    update velocity using Assortative Mating for each particle
    Update particle's position using equation (9)
    Update pBest for each particle in each Task
    Update gBest in each Task
    return the features subset corresponding to gBest that gives us the best accuracy
Function `calculateFitnessValue`:
  Remove all features whose coordinates equals to 0
  Calculate accuracy using a classifier
  Calculate percentage of features selection using equation (13)
  Calculate fitness value using equation (12)
  return fitnessValue

Dataset

In this implementation, we used 16 data sets of height dimensions and different types: Biomedical, Face/Image, and Text. This data can be obtained through the following links: https://ckzixf.github.io/dataset.html, https://jundongl.github.io/scikit-feature/OLD/datasets_old.html and http://tunedit.org/repo/Data/Text-wc.

Table 1 shows the description of the data sets. We experimented data sets No. 2, 3, 4, 5, 6, 7, 11, 12 and 15 shown in the table 1 using the KNN classifier and compared the results with the following PSO-based FS: standard PSO, CSO (Cheng et al., 2015), AMSO (Tran et al., 2019), VLPSO (Tran et all., 2019), PSO-EMT (Chen et al., 2020). Data set No. 1, 8, 9, 10, 13 and 16 are used by a Naïve Bayes classifier. The results were compared to the following Filter approaches: Fast (Song et al., 2013), FCBF (Yu et al., 2003), CFS (Zhao et al., 2007), ReliefF (Robnik-Sikonja et all., 2003), Consist (Dash et al., 2003), FOCUS-SF (Almuallim et al., 1994). We used two different classifiers to give more credibility to our results. Since the authors of PSO-based FS algorithms used the KNN

Figure 5. General view of our approach



classifier in their experiments, we also used it so that we could compare the results. The same is true for the Naive Bayes classifier.

Setting Parameters

Table 2 shows the values for the different settings used in our algorithm.

Results

To examine the performance of the proposed and compared methods, the following metrics are used: 1) the best classification accuracy (Best), the average classification accuracy (Mean), and the standard deviation (Std) based on 20 of the independent runs; 2) the average size of the feature subset (Size); and 3) the average training time (Time).

1. **PHFS-EMT versus PSO-based FS approaches:** Table 3 shows a comparison between the results of PHFS-EMT with other PSO-based FS approaches using the KNN classifier (Since KNN was used in these approaches, we also used it for comparison.). In terms of better accuracy (Best metric), the results show a clear outperformance of the proposed algorithm across all ten data sets. So that we got an accuracy equal to 100% in 5 datasets, which are: Leukemia 1, Leukemia 2

Table 1. Dataset description

No.	Dataset	#F	#I	#Class	Domain
1	PIE10P	2420	210	10	Image
2	Leukemia 1	5327	72	3	Biomedical
3	DLBCL	5469	77	2	Biomedical
4	9Tumor	5726	60	9	Biomedical
5	Brain Tumor 1	5920	90	5	Biomedical
6	Prostate Tumor	5966	102	2	Biomedical
7	Leukemia 2	7129	72	4	Biomedical
8	Tr21.wc	7902	336	6	Text
9	PIXRAW10P	10000	100	10	Image
10	ORL10P	10304	100	10	Image
11	Brain Tumor 2	10367	50	4	Biomedical
12	Leukemia 3	11225	72	3	Biomedical
13	La2s	12432	3075	6	Text
14	11 Tumor	12533	174	11	Biomedical
15	Lang Cancer	12600	203	5	Biomedical
16	La1s	13195	3204	6	Text

Table 2. Parameters setting

Parameter	Explanation	Setting
SWARM SIZE	Number of particles.	100
Max Iteration	The number of iterations used in the BPSO.	100
$C_1 = C_2 = C_3$	The constants C_1, C_2, C_3 are also referred to as trust parameters, where C_1 expresses how much confidence a particle has in itself, while C_2 expresses how much confidence a particle has in its neighbors. Same for C_3.	2
w	Inertia weight is the deciding factor in the convergence behaviour of the PSO. Large inertia weight results in delayed convergence of the optimization and low inertia weight will result in local trapping.	0.5
V_{min}	Minimum limited velocity.	-6
V_{max}	Maximum limited velocity.	+6
α	A parameter that we propose in the fitness function to find a balance between the accuracy of the classification and the number of features selected.	0.2
$\theta 1,\ \theta 2$	Threshold weight value for selecting relevant features from the original features.	0.6

DLBCL, Brain Tumor 2 and Leukemia 3. The lowest accuracy obtained was 90% for the 9_Tumor dataset. But if we look at the accuracy of the original data set of 9_Tumor (Full), we find that the accuracy changed from 40% to 90%, which is a very excellent improvement.

Table 3. Comparison of PHFS-EMT with PSO-based FS approaches using KNN classifier

Dataset	Method	Time (m)	Size	Best	Mean ± std	Dataset	Method	Time (m)	Size	Best	Mean ± std
Leukemia_1	**Full**		5327,00	86,11		Leukemia 2	**Full**		7129,00	81,94	
	PSO	41,20	2615,50	87,36	80,60 ± 2,55		**PSO**	66,09	3513,80	84,58	78,61 ± 2,02
	CSO	247,29	170,12	95,89	90,79 ± 2,88		**CSO**	445,99	389,40	87,50	80,53 ± 2,28
	AMSO	6,80	51,49	97,64	94,01 ± 1,58		**AMSO**	9,66	71,54	89,17	87,52 ± 2,00
	VLPSO	6,09	54,70	97,92	93,31 ± 2,34		**VLPSO**	8,96	**53,39**	92,50	85,82 ± 2,96
	PSO-EMT	9,28	198,40	95,71	91,11 ± 2,79		**PSO-EMT**	12,19	224,44	94,46	90,07 ± 2,47
	PHFS EMT	4,63	22,90	100,00	99,86 ± 0,43		PHFS-EMT	8,50	64,90	100,00	99,03 ± 0,79
DLBCL	**Full**		5469,00	85,71		Brain Tumor 2	**Full**		10367,00	68,00	
	PSO	47,59	2681,00	86,33	83,67 ± 1,52		**PSO**	80,50	5117,20	67,08	61,99 ± 2,91
	CSO	389,67	30,08	**100,00**	94,60 ± 3,26		**CSO**	945,70	90,43	92,00	80,73 ± 5,62
	AMSO	8,34	50,56	96,67	94,10 ± 1,95		**AMSO**	12,06	62,08	81,67	74,96 ± 3,48
	VLPSO	7,18	48,14	93,33	86,51 ± 2,88		**VLPSO**	11,76	81,46	73,33	66,78 ± 4,10
	PSO-EMT	7,02	83,55	**100,00**	93,76 ± 2,80		**PSO-EMT**	**11,51**	499,69	80,00	72,27 ± 4,09
	PHFS EMT	5,28	9,15	100,00	100,00 ± 0,00		PHFS-EMT	13,11	28,65	100,00	97,20 ± 1,99
9_Tumor	**Full**		5726,00	40,00		Leukemia 3	**Full**		11225,00	87.50	
	PSO	39,18	2811,90	45,00	42,72 ± 1,42		**PSO**	120,64	5535,70	92,22	89,83 ± 1,00
	CSO	370,40	220,34	68,33	59,78 ± 3,55		**CSO**	1837,91	88,64	97,30	91,49 ± 3,84
	AMSO	5,52	**52,16**	56,67	50,11 ± 3,61		**AMSO**	15,64	57,19	96,67	94,45 ± 1,04
	VLPSO	5,65	47,05	61,67	54,94 ± 4,80		**VLPSO**	15,94	35,23	94,44	91,56 ± 1,67
	PSO-EMT	8,09	263,09	66,67	58,00 ± 4,02		**PSO-EMT**	**14,72**	268,08	97,32	94,51 ± 1,50
	PHFS EMT	4,44	134,20	90,00	81,47± 3,61		PHFS-EMT	22,16	29,55	100,00	99,72 ± 0,57
Brain Tumor 1	**Full**		5920,00	86,67		11 Tumor	**Full**		12533,00	75,86	
	PSO	66,65	2917,20	77,08	73,73 ± 2,21		**PSO**	4815,54	6205,00	75,59	71,81 ± 1,75
	CSO	457,24	207,61	89,33	80,41 ± 3,93		**CSO**	6278,54	589,36	87,45	83,50 ± 1,70
	AMSO	11,65	93,54	80,00	72,67 ± 3,79		**AMSO**	91,22	319,00	85,06	83,10 ± 1,31
	VLPSO	9,55	**26,83**	79,17	71,19 ± 3,52		**VLPSO**	67,41	249,30	85,21	80,92 ± 2,39
	PSO-EMT	15,43	351,21	90,00	87,37 ± 1,5		**PSO-EMT**	106,53	541,45	89,09	86,15 ± 1,45
	PHFS EMT	7,19	44,95	96,67	95,33 ± 0,85		PHFS-EMT	64,37	280,10	94,25	92,01 ± 1,05
Prostate	**Full**		5966,00	84.31		Lung	**Full**		12600,00		
	PSO	78,77	2926,60	87,33	84,50 ± 1,64		**PSO**	574,17	6234,70	82,72	7877 ± 1,53
	CSO	410,83	207,98	86,14	79,95 ± 3,18		**CSO**	5419,71	230,41	93,47	88,94 ± 1,75
	AMSO	14,31	44,36	94,17	89,58 ± 1,35		**AMSO**	255,32	193,47	92,64	89,97 ± 1,80
	VLPSO	11,05	**35,97**	93,33	88,74 ± 2,33		**VLPSO**	78,00	176,00	92,86	89,55 ± 1,68
	PSO-EMT	16,88	149,86	92,36	89,65 ± 1,82		**PSO-EMT**	134,59	617,61	93,55	91,09 ± 0,94
	PHFS-EMT	8,31	39,15	99,02	97,30 ± 1,18		PHFS-EMT	76,00	143,84	98,03	97,28 ± 0,53

Table 4. Comparison of PHFS-EMT with Filter-based FS approaches using Naive Bayes classifier

Dataset	Method	Time (s)	Size	Best	Mean ± std	Dataset	Method	Time (s)	Size	Best	Mean ± std
PIE10P	**Full**		2420	87,62		orlraws10P	**Full**		10304	86,00	
	Fast	**0,68**	26	96,83			**Fast**	**2,33**	269	99,00	
	FCBF	1,22	48	95,71			**FCBF**	12,99	284	98,80	
	CFS	77,58	61	94,57			**CFS**	30383,93	284	95,60	
	ReliefF	7,64	2202	93,97			**ReliefF**	11,55	10301	94,33	
	Consist	4,19	6	73,9			**Consist**	13,91	4	84,60	
	FOCUS-FS	2,91	6	93,9			**FOCUS-FS**	5,78	4	84,60	
	PHFS-EMT	105,27	28,20	97,62	96,24 ± 0,83		PHFS-EMT	455,70	13,00	**100,00**	99,25 ± 0,79
Tr21.wc	**Full**		7902	46,43		la2s	**Full**		12432	76,29	
	Fast	4,662	8	**89,97**			**Fast**	**70,78**	19	69,68	
	FCBF	5,543	17	48,26			**FCBF**	99,02	41	60,48	
	CFS	218,436	29	53,61			**CFS**	2999,24	67	71,89	
	ReliefF	**4,572**	161	70,43			**ReliefF**	118,62	11	40,99	
	Consist	89,761	16	44,04			**Consist**	7400,29	46	61,94	
	FOCUS-FS	55,644	16	44,04			**FOCUS-FS**	NA	NA	NA	
	PHFS-EMT	435,12	6,77	82,14	78,30 ± 2,49		PHFS-EMT	2451,36	151,75	**80,16**	78,76 ± 0,55
Pixraw10P	**Full**		10000	96,00		la1s	**Full**		13195,00	74,56	
	Fast	**2,96**	15	98,00			**Fast**	**79,62**	22,00	71,96	
	FCBF	9,06	304	98,20			**FCBF**	107,11	46,00	60,46	
	CFS	25372,21	235	96,80			**CFS**	2668,87	67,00	70,42	
	ReliefF	11,23	10000	98,00			**ReliefF**	128,45	**8,00**	33,51	
	Consist	9,88	3	90,00			**Consist**	7830,98	45,00	58,74	
	FOCUS-FS	4,29	3	90,00			**FOCUS-FS**	NA	NA	NA	
	PHFS-EMT	424,43	3,70	**100,00**	100,00 ± 0,00		PHFS-EMT	3274,24	226,05	**81,71**	80,32 ± 0,82

We note that the classification accuracy of the selected feature subsets was better than the accuracy of the original feature set. We note that the value of the standard deviation (std) was close to 0 in most datasets (it reached 0 in DLBCL), which confirms that the search process was well oriented by the proposed fitness function.

The fourth column (size) represents the average number of selected features (for the 20-independence run). We notice that the data size has decreased from thousands of features to a

few features. For example, in the Lung Cancer dataset, the number of features shifted from 12 600 to 143.84. Compared with other algorithms, we find that our algorithm got the least number of features in 6 datasets.

The execution time of PHFS-EMT and the compared approaches are shown in the third column of Table 3 (Time). The execution time of PHFS-EMT is the smallest among all the six methods on seven datasets (i.e., Leukemia 1, DLBCL, Brain Tumor 1, Prostate, Leukemia 2, 11_Tumor, Lung_cancer). Compared with PSO, the training time of PHFS is between 1/5 and 1/9 of that of original PSO. CSO needs the longest computational time among all compared algorithms. The principal reason is that CSO needs to save the fitness values of the previously chosen feature subsets to avoid repeated assessments. However, the fitness evaluation time saved appears to be influenced by the time required to match the archiving solution. Compared with AMSO, VLPSO and PSO-EMT, the training time of PHFS is a little longer on two datasets (Brain Tumor 2, Leukemia 3). This is because a feature ranking method was embedded in AMSO and VLPSO to create solution representation strategies, which can effectively decrease the search space during the FS process.

In general, among the six algorithms the proposed PHFS method has the best tradeoff between the effectiveness and efficiency, whether in terms of classification accuracy, number of features, or execution time.

2. **PHFS-EMT Versus Filter methods:** Table 4 shows a comparison result between our algorithm and other Filter-based FS algorithms. It is known that the Filter methods are characterized by the speed of execution, so we note that our algorithm was less fast compared to other approaches, especially the Fast algorithm, which was the fastest in most cases. The average number of selected features obtained was very improved compared to the total number of features. As for the classification accuracy, we obtained the best accuracy compared to other algorithms in 5 datasets out of 6. We obtained an accuracy equal to 100% in two datasets, namely Pixraw10P and orlraws10P. The lowest accuracy we obtained was 80.16%. Fast algorithm obtained the highest accuracy for Tr21.wc dataset, estimated at 89.97%, while our algorithm obtained 82.14%. We note that the classification accuracy of the selected feature subset was better than the accuracy of the original feature set. The standard deviation (std) was close to 0 in most datasets (it reached 0 in Pixraw10P), which confirms that the search process was well oriented by the proposed fitness function.

PHFS-EMT running time: Filter algorithms are known to be very fast, so we note that the Fast algorithm was the fastest in all six groups, FCBF was also fast. PHFS-EMT recorded very reasonable times, given the size of the data, the largest time took 2767.12 seconds, equivalent to 46.12 minutes, and that was in the l1as data set. The smallest time took 105.27 seconds, equivalent to 1,75 minutes.

Table 5 shows a comparison result between PHFS-EMT and other Filter-based FS algorithms using C4.5 classifier. We note that our algorithm was less fast compared to other approaches, especially the Fast algorithm, which was the fastest in most cases. The average number of selected features obtained was very improved compared to the total number of features. As for the classification accuracy, we obtained the best accuracy compared to other algorithms in all datasets. The standard deviation (std) was close to 0 in most datasets (it reached 0 in Pixraw10P), which confirms that the search process was well oriented by the proposed fitness function.

3. **Results of each task:** Table 6 shows the number of times the problem was solved by Task 1 or Task 2, for each dataset (for the 20 independent runs). We note that Task 1 was selected as a solution 115 times and Task 2 was selected 205 times. We note that in the medical data and images, Task 2 was mostly the solution. Whereas in the text Task 1 was the solution. Here's the question: Is there an association between the correlation measure and the data type? The results show that the Symmetric Uncertainty measure was more effective with textual data, while the

Table 5. Comparison of PHFS-EMT with Filter-based FS approaches using C4.5 classifier

Dataset	Method	Time (s)	Size	Best	Mean ± std	Dataset	Method	Time (s)	Size	Best	Mean ± std
PIE10P	Full		2420	77,62		orlraws10P	Full		10304	71,00	
	Fast	0,68	26	84,13			Fast	2,33	269	90,33	
	FCBF	1,22	48	82,95			FCBF	12,99	284	82,60	
	CFS	77,58	61	86,1			CFS	30383,93	284	90,20	
	ReliefF	7,64	2202	80,32			ReliefF	11,55	10301	74,67	
	Consist	4,19	6	81,33			Consist	13,91	4	84,00	
	FOCUS-FS	2,91	6	81,33			FOCUS-FS	5,78	4	84,00	
	PHFS-EMT	237,42	32,00	91,9	90,95 ± 1,35		PHFS-EMT	455,70	13,00	99,00	97,50 ± 1,05
Tr21.wc	Full		7902	81,85		la2s	Full		12432	76,29	
	Fast	4,662	8	90,47			Fast	70,78	19	69,68	
	FCBF	5,543	17	88,32			FCBF	99,02	41	60,48	
	CFS	218,436	29	69,23			CFS	2999,24	67	71,89	
	ReliefF	4,572	161	80,26			ReliefF	118,62	11	40,99	
	Consist	89,761	16	68,76			Consist	7400,29	46	61,94	
	FOCUS-FS	55,644	16	88,21			FOCUS-FS	NA	NA	NA	
	PHFS-EMT	358,60	32,00	95,24	94,23 ± 1,06			5970,98	111,2	81,79	80,22 ± 0,32
Pixraw10P	Full		10000	94,00		la1s	Full		13195,00	76,78	
	Fast	2,96	15	97,00			Fast	79,62	22,00	72,45	
	FCBF	9,06	304	95,40			FCBF	107,11	46,00	72,57	
	CFS	25372,2	235	98,80			CFS	2668,87	67,00	77,63	
	ReliefF	11,23	10000	93,33			ReliefF	128,45	8,00	47,05	
	Consist	9,88	3	92,20			Consist	7830,98	45,00	75,33	
	FOCUS-FS	4,29	3	92,20			FOCUS-FS	NA	NA	NA	
	PHFS-EMT	354,86	3,83	99,00	99,00 ± 0,00		PHFS-EMT	6120,23	220,33	81,65	81,37 ± 0,40

coefficient of Pearson was more effective with medical data and images. But the answer to the previous question requires deep study. We mentioned earlier that evolutionary multitasking relies on information shared between tasks. So, table 7 shows the number of common features between Task 1 and Task 2 in our experimentations. We note that there are a good number of common features for all datasets.

4. **Classification Performance:** Accuracy is a metric to measure the degree of efficiency of the learning model, which we used to evaluate our results, but there are other important metrics, including: TP Rate (True Positive Rate) is used to measure the percentage of actual positives that were correctly identified, Precision used for determine the number of positive class predictions that actually belong to the positive class, Recall used for determine the number of positive class predictions made out of all positive examples in the dataset, and F-Measure is a single score that balances both the concerns of precision and recall in one number, The MCC (Matthews correlation coefficient) is essentially a correlation coefficient between observed and predicted binary classifications, it returns a value between -1 and $+1$. A coefficient of $+1$ represents a perfect prediction, 0 no better than the random prediction and -1 indicates a total disagreement between the prediction and the observation. To confirm the efficiency of our proposed model, we used these metrics. Table 8 shows the results. Note that most values are close to or equal to 1, while **FP Rate** values are close to or equal to 0, which confirms the efficiency of our proposed model.
5. **Parallel execution vs sequential execution:** To evaluate the effectiveness of the proposed parallel processing, we implemented our algorithm in sequential mode as well (without Apache Spark). Figure 6 shows the graph of the execution time in parallel and serial mode for six datasets with Naïve Bayes classifier.

Knowing that the blue bar represents the sequential time, and the red bar represents the parallel time. We note that the gained time (the difference between the serial and the parallel time) increases with the increase in the serial time, so that the execution time for the l1as.wc dataset in the serial mode was 304.31 minutes, while in the parallel mode it was 54.57 minutes, that is, we gained 249.74 minutes.

Table 6. The number of times the problem was solved by a task

Dataset	Solved By		Number of independences runs
	#Task1	#Task2	
Leukemia_1	0	20	20
DLBCL	0	20	20
9_Tumor	2	18	20
Brain Tumor 1	5	15	20
Prostate	0	20	20
Leukemia 2	0	20	20
Brain Tumor 2	0	20	20
Leukemia 3	6	14	20
11 Tumor	20	0	20
Lung	0	20	20
PIE10P	0	20	20
Tr21.wc	2	18	20
Pixraw10P	20	0	20
orlraws10P	20	0	20
la2s	20	0	20
la1s	20	0	20
Total	**115**	**205**	**320**

Figure 7 shows the graph of the execution time in parallel and sequential mode for biomedical datasets with KNN classifier. Knowing that the blue bar represents the parallel time, and the red bar represents the sequential time. For the LangCancer data set, the sequential time was 91.88 minutes, and the parallel time was 76.00minutes, meaning we gained 15.88 minutes. Also, for 11 Tumor, the gained time was 12.84 minutes. The least time gained was 3.247 minutes for the DLBCL dataset. We conclude from the above that the parallel fitness evaluation of particles in the PSO algorithm; which we have proposed; was effective.

Note that the time gained using KNN classifier was not great compared to the time gained using Naive Bayes. It concludes that the quality of the classifier has an impact on the speed of execution. We also note that the number of features is not the only factor that increases the complexity of the execution time, but there is also the number of instances.

Figure 8 shows the graph of the execution time in parallel and serial mode for six datasets with C4.5 classifier. Knowing that the blue bar represents the sequential time, and the red bar represents the parallel time. We note that the gained time (the difference between the serial and the parallel time) increases with the increase in the serial time, so that the execution time for the tr21.wc dataset in the serial mode was 444.21 minutes, while in the parallel mode it was 5.95 minutes, that is, we gained 438,23 minutes.

However, from these results, we can conclude that the parallel architecture that we proposed was effective.

CONCLUSION

In this paper, we have addressed the problem of High data dimension in the context of big data, and its negative impact on classification algorithms. To solve this problem, we proposed a hybrid feature

Table 7. Common features

Dataset	#Common Features
la1s.wc	215
la2s.wc	186
tr21.wc	274
Prostate Tumor 1	453
DLBCL	306
Lung Cancer	575
Leukemia 1	296
Brain Tumor 1	318
Leukemia 3	673
Leukemia 2	404
PIE10P	31
Brain Tumor 2	396
11 Tumor	772
orlraws10P	88
9 Tumor	271
Pixraw10P	194

Table 8. Classification performance

Dataset	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	Proc Area	PRC Area
la1s.wc	0,817	0,043	0,813	0,817	0,813	0,772	0,939	0,816
la2s.wc	0,802	0,045	0,804	0,802	0,800	0,755	0,938	0,820
tr21.wc	0,821	0,356	0,801	0,821	0,800	0,768	0,750	0,743
Prostate Tumor 1	0,990	0,009	0,990	0,990	0,990	0,981	0,998	0,996
DLBCL	1,000	0,000	1,000	1,000	1,000	1,000	1,000	1,000
Lung Cancer	0,980	0,033	0,981	0,980	0,980	0,958	0,974	0,966
Leukemia 1	1,000	0,000	1,000	1,000	1,000	1,000	1,000	1,000
Brain Tumor 1	0,967	0,067	0,968	0,967	0,964	0,932	0,950	0,937
Leukemia 3	1,000	0,000	1,000	1,000	1,000	1,000	1,000	1,000
Leukemia 2	1,000	0,000	1,000	1,000	1,000	1,000	1,000	1,000
PIE10P	0,976	0,003	0,978	0,976	0,976	0,974	0,989	0,981
Brain Tumor 2	1,000	0,000	1,000	1,000	1,000	1,000	1,000	1,000
11 Tumor	0,943	0,007	0,949	0,943	0,943	0,939	0,964	0,894
orlraws10P	1,000	0,000	1,000	1,000	1,000	1,000	1,000	1,000
9 Tumor	0,900	0,014	0,908	0,900	0,898	0,888	0,943	0,828
Pixraw10P	1,000	0,000	1,000	1,000	1,000	1,000	1,000	1,000

Figure 6. Parallel vs sequential execution using NB classifier

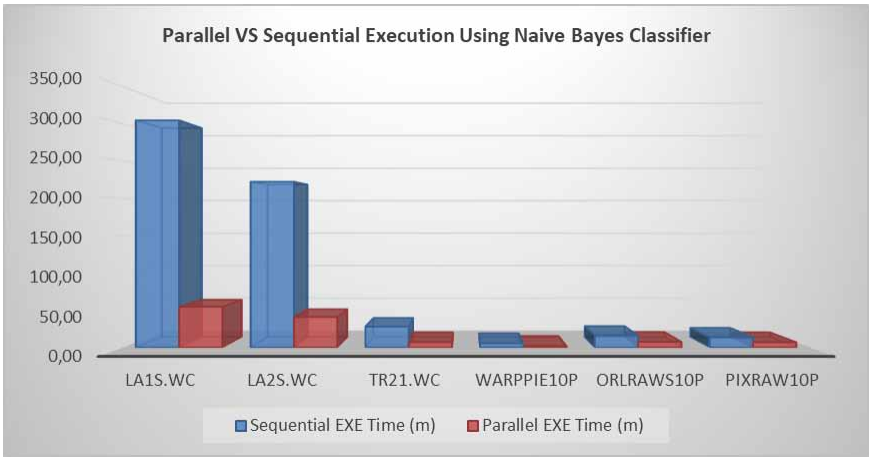


Figure 7. Parallel vs sequential execution using KNN classifier

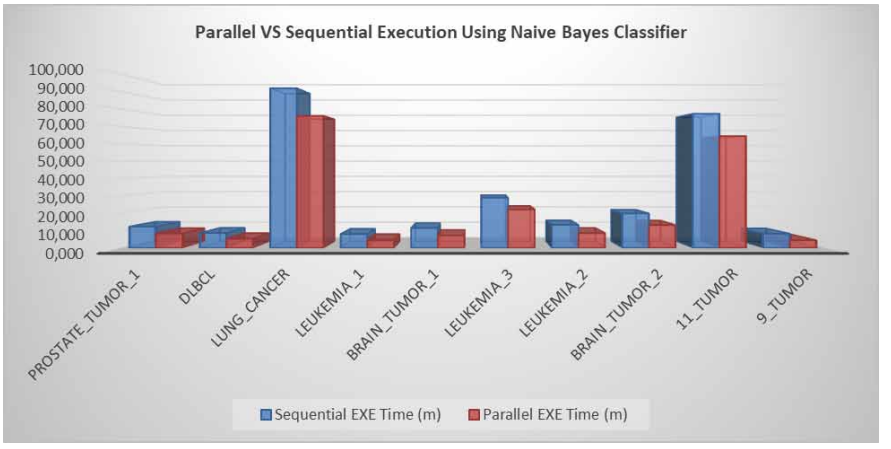


Figure 8. Parallel vs sequential execution using C4.5 classifier

selection algorithm based on multi-correlation and evolutionary multi-tasking. To improve the runtime, we have proposed a parallel correlation computation, and a parallel fitness evaluation of BPSO under Apache Spark. The results showed that employing the concept of evolutionary multitasking with multi-correlations measures contributed to the improvement of classification performance with a significant reduction in the size of the data. The results also showed that the proposed parallel processing contributed to reducing the runtime in a very significant time. The proposed approach suffers somewhat when dealing with data that contains many instances, especially in the runtime. In future work, we will focus on improving these shortcomings.

REFERENCES

Blessie, C. E., & Karthikeyan, E. (2012). A Feature Selection Algorithm Using Correlation Based Method. Journal of Algorithms & Computational Technology, 6(3).

Almuallim, H., & Dietterichb, G. (1994, September). T. (1994). Learning Boolean concepts in the presence of many irrelevant features. Artificial Intelligence, 69(1–2), 279–305.

Banka, H., & Dara, S. (2015). A hamming distance based binary particle swarm optimization (HDBPSO) algorithm for high dimensional feature selection, classification and validation. Pattern Recognition Letters, 52, 94–10.

Beesetti, K., Bilgaiyan, S., & Prasad Mishra, B. (2022). A hybrid feature selection method using multi-objective Jaya algorithm. *2022 International Conference on Computing, Communication and Power Technology (IC3P).*

Ceylan, O., & Taşkın, G. (2021). Feature Selection Using Self Organizing Map Oriented Evolutionary Approach. *2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS.*

Chen, K., Xue, B., Zhang, M., & Zhou, F. (2020). An evolutionary multitasking-based feature selection method for high-dimensional classification. IEEE Transactions on Cybernetics.

Chen, K., Zhou, F., & Yuan, X. (2019). Hybrid particle swarm optimization with spiral-shaped mechanism for feature selection. Expert Systems with Applications, 128, 140–15.

Cheng, R., & Jin, Y. (2015, February). A Competitive Swarm Optimizer for Large Scale Optimization. IEEE Transactions on Cybernetics, 45(2), 191–204.

Dai, J., Wang, W., & Xu, Q. (2013). An uncertainty measure for incomplete decision tables and its applications. IEEE Transactions on Cybernetics, 43(4), 1277–1289.

Dash, M., & Liub, H. (2003). Consistency-based search in feature selection. *Artificial Intelligence, 151*(1), 155-176.

Feng, L., Zhou, W., Zhou, L., Jiang, S. W., Zhong, J. H., Da, B. S., Zhu, Z. X., & Wang, Y. (2017). An empirical study of multifactorial PSO and multifactorial DE. *IEEE Congr. Evol. Comput*., 921–928.

Gu, S., Cheng, R., & Jin, Y. (2018). Feature selection for high-dimensional classification using a competitive swarm optimizer. Soft Computing, 22, 811–822.

Gupta, A., Ong, Y., Feng, L., & Tan, C. (2017, July). Multiobjective multifactorial optimization in evolutionary multitasking. IEEE Transactions on Cybernetics, 47(7), 1652–1665. doi:10.1109/TCYB.2016.2554622 PubMed doi:10.1109/TCYB.2016.2554622 PMID:27164616

Hall, M. A. (1992). *Correlation-Based Feature Subset Selection for Machine Learning* [PhD dissertation]. Univ. of Waikato.

Holmes, G., Donkin, A., & Witten, I. H. (1994). WEKA: a machine learning workbench. *Proceedings of ANZIIS '94 - Australian New Zealnd Intelligent Information Systems Conference.*

Jovi'c, A., Brki'c, K., & Bogunovi´, N. (2015, May). A review of feature selection methods with applications. MIPRO, 2015, 25–29. doi:10.1109/MIPRO.2015.7160458 doi:10.1109/MIPRO.2015.7160458

Kennedy, J., & Eberhart, R. C. (1997). A discrete binary version of the particle swarm algorithm. *IEEE 1997 International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation, 5,* 4104-4108.

Lane, C., Xue, B., Liu, I., & Zhang, M. (2014). Gaussian based particle swarm optimization and statistical clustering for feature selection. *Proc. Eur. Conf. Evol. Comput. Combinatorial Optim.*, 133—144.

Lane, M., Xue, B., Liu, I., & Zhang, M. (2014). Gaussian based particle swarm optimization and statistical clustering for feature selection. *Proc. Eur. Conf. Evol. Comput. Combinatorial Optim.*, 133–144.

Lui, X., & Shang, L. (2013). A Fast wrapper feature subset selection method based on binary particle swarm optimization. *IEEE Congr. Evol. Comput.,* 3347—3353.

Nugroho, A., Zainul Fanani, A., & Shidik, G. (2021). Evaluation of Feature Selection Using Wrapper for Numeric Dataset with Random Forest Algorithm. *2021 International Seminar on Application for Technology of Information and Communication (iSemantic).*

Rong, M., & Gong, D., & Gao, X. (2019). Feature Selection and Its Use in Big Data: Challenges, Methods, and Trends. *IEEE Access, 7,* 19709-19725.

Shaikh, E., Mohiuddin, I., Alufaisan, Y., & Nahvi, I. (2019). *Apache Spark: A Big Data Processing Engine.* IEEE.

Song, Q., Ni, J., & Wang, G. (2013). A Fast Clustering-Based Feature Subset Selection Algorithm for High-Dimensional Data. IEEE Transactions on Knowledge and Data Engineering, 25(1), 1–14. doi:10.1109/TKDE.2011.181 doi:10.1109/TKDE.2011.181

Too, J., Rahim Abdullah, A., & Mohd Saad, N. (2019). A new co-evolution binary particle swarm optimization with multiple inertia weight strategy for feature selection. Informatics (MDPI), 6(2), 21.

Tran, B., Xue, B., & Zhang, M. (2019). Adaptive multi-subswarm optimization for feature selection on high-dimensional classification. *GECCO '19 Proceedings of the Genetic and Evolutionary Computation Conference,* 481–489.

Tran, B., Xue, B., & Zhang, M. (2019, June). Variable-length particle swarm optimization for feature selection on high-dimensional classification. IEEE Transactions on Evolutionary Computation, 23(3), 473–487.

Tran, B., Zhang, M., & Xue, B. (2016). A PSO based hybrid feature selection algorithm for high-dimensional classification. *IEEE Congress on Evolutionary Computation (CEC)*, 3801-3808.

Tran, B., Zhang, M., & Xue, B. (2016). A PSO based hybrid feature selection algorithm for high-dimensional classification. *IEEE Congr. Evol. Comput.,* 3801—3808.

Xue, B., Zhang, M., & Browne, W. (2013, December). Particle swarm optimization for feature selection in classification: A multi-objective approach. IEEE Transactions on Cybernetics, 43(6), 1656–1671.

Xue, B., Zhang, M., Browne, W., & Yao, X. (2016). A survey on evolutionary computation approaches to feature selection. IEEE Transactions on Evolutionary Computation, 20(4), 606–626.

Yu, L., & Liu, H. (2003). Feature Selection for High-Dimensional Data: A Fast Correlation Based Filter Solution. *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)*, 856–863.

Yu, W., Kang, H., Sun, G., Liang, S., & Li, J. (2022). Bio-Inspired Feature Selection in Brain Disease Detection via an Improved Sparrow Search Algorithm. *IEEE Transactions on Instrumentation and Measurement, 72.*

Zhao, Z., & Liu, H. (2007). Searching for interacting features. *Proceedings of International Joint Conference on Artificial Intelligence*, 1156–1161.

Mohamed Amine Azaiz received the engineer degree in computer science from Djillali Liabes University, Sidi Bel Abbes in 2008, and the Master degree in computer science from Djillali Liabes University in 2015. Actually, He is currently the Ph.D. degree in Data and Software Engineering at the High School of computer sciece of Sidi Bel Abbes, Algeria.

Djamel Amar Bensaber Received the Ph.D. degree in computer science and HDR from Djillali Liabes University, Sidi Bel Abbes in 2008 and 2014, respectively. Actually, he is a Full Professor at the High School of computer science of Sidi Bel Abbes, Algeria. He is currently Head of Research Team 'Data and Software Engineering' at the LabRI Laboratory. His research interests include Information Systems (Business Informatics), Software Engineering, Big data; Linked open data, Ontology, Semantic Computing, Data & Knowledge Management; Model-driven engineering.