Neural Computing and Applications (2020) 32:7839–7857
https://doi.org/10.1007/s00521-019-04171-3

A wrapper-filter feature selection technique based on ant colony optimization

Manosij Ghosh[1] · Ritam Guha[1] · Ram Sarkar[1] · Ajith Abraham[2]

Received: 19 November 2018 / Accepted: 29 March 2019 / Published online: 11 April 2019

Abstract
Ant colony optimization (ACO) is a well-explored meta-heuristic algorithm, among whose many applications feature selection (FS) is an important one. Most existing versions of ACO are either wrapper based or filter based. In this paper, we propose a wrapper-filter combination of ACO, where we introduce subset evaluation using a filter method instead of using a wrapper method to reduce computational complexity. A memory to keep the best ants and feature dimension-dependent pheromone update has also been used to perform FS in a multi-objective manner. Our proposed approach has been evaluated on various real-life datasets, taken from UCI Machine Learning repository and NIPS2003 FS challenge, using K-nearest neighbors and multi-layer perceptron classifiers. The experimental outcomes have been compared to some popular FS methods. The comparison of results clearly shows that our method outperforms most of the state-of-the-art algorithms used for FS. For measuring the robustness of the proposed model, it has been additionally evaluated on facial emotion recognition and microarray datasets.

Keywords Wrapper-filter method · Ant colony optimization · Feature selection · NIPS2003 challenge

1 Introduction

A typical pattern recognition problem often uses many features in order to achieve high recognition accuracy. But, many times, it is seen that only some of the used features have the discriminatory power to classify the patterns accurately. Therefore, for most of the real-world problems, researchers introduce many candidate features causing an increase in feature dimension. However, the irrelevant or redundant features reduce the performance of the learning system both in terms of speed (due to the high dimension of the feature vector) and recognition accuracy (due to the presence of inapt information). This is referred to as the 'curse of dimensionality' [1]. This situation becomes more serious when researchers deal with time series data or gene expression data, as huge number of candidate features are used to represent a sample. For example, in gene expression data [2], the number of genes which are biomarkers in gene expression data is very small in comparison with the large dimension of the data.

Hence, data preprocessing has become almost necessary to reduce the noise and improve the quality of recognition system. There are many data preprocessing techniques to fine tune the same before its use. Among these techniques, feature dimension reduction or feature selection (FS) is a very important one. When datasets become so complex and huge, FS methods to refine the information and restrict them only to the relevant and useful features are of great importance. As FS aims to lessen the computational time and cost, this preprocessing technique is very valuable in the field of data analysis.

Broadly FS algorithms can be divided into three categories *namely* filter methods, wrapper methods and

✉ Ram Sarkar
rsarkar@cse.jdvu.ac.in; raamsarkar@gmail.com

Manosij Ghosh
manosij1996@gmail.com

Ritam Guha
ritamguha16@gmail.com

Ajith Abraham
ajith.abraham@ieee.org

[1] Computer Science and Engineering Department, Jadavpur University, 188, Raja S.C. Mallick Road, Kolkata, West Bengal 700032, India

[2] Scientific Network for Innovation and Research Excellence, Machine Intelligence Research Labs (MIR Labs), Auburn, WA 98071, USA

wrapper-filter or embedded methods. Filter methods work on the intrinsic properties of data and do not require a learning algorithm. This tends to make them very fast but as FS is done without consultation of a learning algorithm, the accuracy of FS using filter methods is generally less than wrapper methods. A few examples include ReliefF [3], Similarity measure [4], Gini index [5], etc. Wrapper methods require a learning algorithm which leads to a higher accuracy but also a much higher computation time. Some popular methods of this category are genetic algorithm (GA) [6], particle swarm optimization (PSO) [7], ant colony optimization (ACO) [8], etc. A compromise between these two methods is embedded methods which are built using a combination of both filter and wrapper methods. These techniques strike a balance between the two classes of methods and try to incorporate the use of learning algorithms as well as intrinsic data properties in a method. There may be a fine trade-off between computation time and accuracy or even a lower computation cost with no accuracy degradation. Therefore, the general trend has moved to the design of embedded systems. A few examples of this class of algorithms include memetic algorithm (MA) [9], Markov blanket-embedded genetic algorithm (MBEGS) [10], etc.

Nature has inspired the researchers to devise new algorithms which are used to solve complex optimization problems. Various natural elements and their unique way of survival have attracted us over the years. Natural phenomenon like metal annealing too has inspired an algorithm—simulated annealing (SA) [11]. Till date, many complicated FS algorithms have been derived from swarms' unique techniques to gather food for survival or escape from predators. These algorithms are commonly named as 'Swarm Intelligence Optimization Algorithms.' Some popular swarm-based algorithms are ACO, PSO, Cuckoo search algorithm (CSA) [12], artificial bee colony (ABC) [13], bee colony optimization feature selection (BCOFS) [14], etc.

In recent times, the behavior of animals has inspired a number of meta-heuristic algorithms. While some inspirations come from large mammals like whale which has lead to the development of whale optimization algorithm (WOA) [15], other motivations come from relatively smaller mammals like grey wolf called grey wolf optimization (GWO) [16] and even the flying squirrel (squirrel search algorithm) [17]. Also, microbes have managed to inspire algorithms like virus colony search [18] which is modeled on the diffusion and infection strategies adopted by virus colonies to survive and spread in host cells. A number of these approaches have been used for performing FS. Like WAO adapted for application to FS by using crossover and mutation [19]. It has been hybridized with SA to perform FS [20]. GWO too has been used for FS in [21]. A few classical methods too have also been modified in recent times to enhance their performance. Mutation-enhanced PSO (ME-PSO) [22] is proposed to enhance results of PSO using mutation and memory renewal to store subsets. GA was modified to form histogram-based multi-objective GA (HMOGA) [23] where multiple runs of GA are combined using a histogram to yield the best subset. Bee colony optimization is also adapted for application to FS [14].

Ant colony optimization or ACO is a popular optimization algorithm which is conceptualized from the food searching technique of real-life ants. Ants are well-known natural elements following a specific pattern while searching for food from their nest. Now the question is how does this specific pattern come into action? The answer is a special chemical known as *pheromone* which is the key due to which ants follow a special pattern. In nature, ants target the shortest path between their nest and the food source without the use of any visual information. When an ant moves through any path, it deposits some amount of pheromone on that path. Gradually pheromone evaporates, thereby decreasing the intensity of pheromone on all the paths. Evaporation is an essential phenomenon which helps to get rid of the problem of getting stuck in a local minima and to increase exploration of the search space. Therefore, level of pheromone can clearly determine the popular paths traversed by the ants and so they tend to follow the path with more pheromone content. Thus, gradually the shortest path tends to be traversed by them more than the other paths.

In ACO, the ants are considered as stochastic procedures which build the feature subsets. Iteratively, subsets are generated using both heuristic information as well as the pheromone trails created by the ants, i.e., in ACO knowledge is acquired dynamically. The stochastic component brings about a more thorough exploration of the search space and creates a wide variety of subsets compared to greedy heuristics. Heuristic information guides ants toward more promising subsets. The search strategy of ants is reminiscent of reinforcement learning [24]. The fact that FS is done using a colony of ants helps in greater parallel search as well as sharing of information through pheromones. This has motivated the researchers to design ACO algorithm which has been used to solve various optimization problems till date.

Our work focuses on building a multi-objective FS algorithm based on ACO. Though ACO was primarily developed for solving traveling salesman problem, it can be customized to fit into FS domain. From the literature survey, it has been observed that ACO outperforms many contemporary FS methods both in terms of accuracy and feature reduction. But it is to be noted that ACO's existing versions are mostly filter and wrapper based. These facts motivate us to develop a filter-wrapper version of ACO. We have evaluated our work on many real-life datasets to prove its applicability.

2 Literature study

This idea of ACO was first implemented by Dorigo et al. [25] and they named it as ant system (AS). Since then, many modifications on ACO have taken place over the years. In 1996, it was modified to ant colony system (ACS) [26] for application to traveling salesman problem (TSP). Another modification proposed in 2000 of AS can be found in [27] called max–min ant system which improves exploration by allowing only best ants to add pheromones. This makes the search greedier and shows good results in TSP and quadratic assignment problem. Pheromone updation was modified in [28] to be done in two stages instead of one. First, a number of top solutions deposit the pheromones followed by deposition of pheromone by the optimal solution (ant).

Though mainly used for shortest path discoveries in graphs, in [29] an ACO was proposed for application in text FS problem. A global and local update rules have changed the pheromone trails of the paths after each iteration. Depending on pheromone intensities and heuristic desirability, the path of traversals is chosen. They used a learning algorithm to calculate the heuristic desirability of the traversals. After every iteration, a stopping criterion is checked and if it is not satisfied, then this process starts again with completely new set of ants and it continues till the stopping criterion is fulfilled.

Recently, in [30], authors have suggested an unsupervised FS algorithm based on ACO. In this method, when ants construct solutions, they use a similarity matrix to select the next feature based on similarity between the last selected feature and feature to be selected next. After constructing the solutions, pheromones are updated only based on frequency of selection of features. A similar approach [31] for calculating heuristic desirability is to use the product of similarity of two features and the inverse of the average similarity of all the features not selected. Another filter approach [32] based on similarity and applied to microarray datasets used a fully connected graph and similarity measure as the weights of the edges of the graph. The subsets are built and evaluated using the relevancy of the genes (average of all the relevancy of genes to the class) and tried to maximize it. Though simple, it provides impressive results.

There are various algorithms which use a filter method to find the heuristic desirability. In [33], Fisher score and correlation are used as metrics for heuristic desirability. Also, here authors use a bi-directional circular graph where a path exists to move from one node to another through two paths—one selecting the next node and the other discarding it. This approach makes the traversal very restricted as they enforce an ordering in which features can be selected. Another approach [34] on the same line applies a fully connected graph with each feature as a node and each node consists of two subnodes: a ‘0’ and a ‘1’. A ‘0’ denotes that the feature is discarded and ‘1’ means that the feature is selected.

An innovative approach to FS (using multivariate filter method) using ACO is reported in [35] where the authors cluster the features using Louvain community detection method. All the features are represented as nodes in a fully connected unidirectional graph where each edge is assigned a similarity value based on Pearson coefficient. The ants select features from one cluster and generate a random number to decide if it wants to continue in same cluster or visit an unvisited cluster. An improvement of this algorithm is proposed in [36], where the initial pheromone values are initialized using the relevance of the features. Further, the evaluation function uses multiple discriminatory analysis, while a more effective cost function is also used.

In [37], a new classification algorithm called PMC—Pattern Matching Classification is proposed where the authors have developed an ACO to match their PMC. In each ant, they store a list of features selected by the ant (solution list) and the features not selected by the ant are out in storage list. On each iteration, a random number of features from solution list and a random number from the storage list are taken and the new set is evaluated using leave one out cross-validation (LOOCV). If better, then the storage and solution lists are updated so that the new set is in the solution list and rest are in the storage list. Though innovative, it does not use any heuristic to form the sets which are built randomly and therefore does not give good result.

It is to be noted that the problem of getting stuck in local optima is not addressed by many, but the method described in [8] is an exception in this regard. The use of a local pheromone update allows the algorithm to prevent premature convergence, whereas they also use formulations to decrease pheromone content of frequently traversed edges. Variety of local search mechanisms are developed and inserted in the ACO to help reach better results. However, like most wrapper methods of ACO, it suffers from very high computation time which also restricts the use of costly but useful classifiers, like multi-layer perceptron (MLP) or support vector machine (SVM) [38].

Hybrid algorithms in ACO are almost absent to the best of our knowledge. The one that exists [39] merely incorporates information gain (IG) of a feature in the calculation of heuristic desirability along with a classifier which evaluates the subsets after each feature addition by the ant. Therefore, this leads to a huge computation time like wrapper methods described in [8, 29]. The method uses the product of subset accuracy and the IG of the feature as heuristic desirability. IG does not reflect the ability of a feature to be useful as part of a subset in a particular classifier, rather shows the dependence of the feature on the class labels. Therefore, this model might not provide desirable results at all times.

ACO has also been applied to FS domain. Fallahzadeh et al. [40] have applied ACO-based FS for determining the optimal feature set in classification of breast cancer into normal, benign and cancerous. Features are extracted using Raman spectroscopy from preprocessed images (background noise and background fluorescence are removed). ACO has also been used for FS in diagnosis of pulmonary hamartoma [41]. From CT scan images, regions of interest are identified and from these regions shape, texture and run-length-based features are extracted and ACO is used for FS. While FS is an important application domain for ACO, a number of works have also focused on other application domains. In [42], ACO is utilized for induction motor's auto-disturbance rejection control. This has helped to automatically tune the motor to achieve the exact decoupling. Another application of ACO is found in recommender system [43] where ACO is used to assign closeness to target user. In fact, even in the domain of economics, ACO is used for the prediction of a financial crisis [44]. This goes a long way in ascertaining the popularity and applicability of ACO.

3 Present work

The basis for our proposed method is described in Sect. 3.1 while our proposed method is detailed in Sect. 3.2.

3.1 Basis of proposed method

There are several works on ACO among which we have selected two versions of ACO—one Text FS ACO, abbreviated as TFSACO [29] and another abbreviated as Unsupervised FS ACO (UFSACO) [30]. To build our method, the two versions have been combined to include their advantages as well as address their problems. In ACO, the ants move about in a connected graph like structure and the paths they walk on are marked with pheromones. This structure, however, suffers from the limitation that the same subsets can be built by walking through different paths. For example, as shown in Fig. 1 if we have {A, B, C, D, E} as the nodes (features), the paths which include those in a fully connected graph can be {A to B to C to D to E}, {A to C to D to B to E} among others. This means that

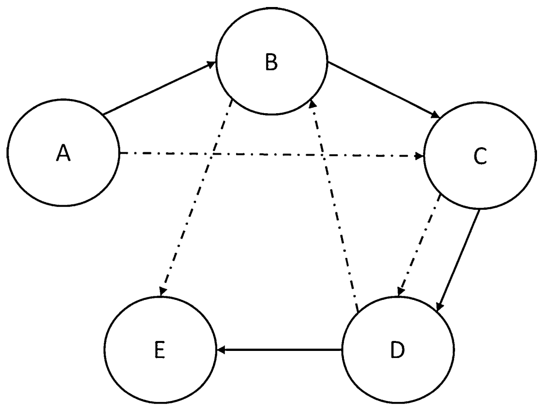

Fig. 1 Showing multiple paths in graph structure which can be used to access the same set of features, two paths containing feature set {A, B, C, D, E} are shown in bold and dashed lines

pheromone deposits do not occur properly. As multiple paths will have pheromone deposits due to the same subset of nodes (exact number of possible paths for a subset of size n is $n!$), so pheromone deposit is distributed unevenly. This problem can be overcome by depositing the pheromones on the node (feature) instead of the path or the edge between the nodes.

In TFSACO and UFSACO, pheromone update is done globally and locally but the value of the pheromone is not bounded. Hence, a feature chosen more times has a very high pheromone value to ensure its multiple selection every time. Eventually, this makes the FS biased and hinders the exploration of all the features. To address this problem, we normalize the pheromone values to ensure a bound $[0, 1]$ after each iteration.

In general, in ACO the ants move in the graph by traveling from one node to another through the edges. On reaching a new node (here, feature) the ants compute the probability of adding new feature to existing list using pheromone density and heuristic desirability. A random non-selected feature (i.e., not in subset) is selected. Heuristic desirability is accuracy of the subset if the feature is added or some measure of goodness of subset. Pheromone density is the pheromone on the path from the last feature added to selected feature or the pheromone on selected feature.

Probability of feature addition, as already discussed, has two measures *namely*, pheromone density and heuristic desirability. The heuristic desirability is calculated (in all wrapper-based ACOs) using a classifier to find the classification ability of the new feature. Therefore, at each step of feature addition, the classifier is used.

The use of classifier for n times (where n is the number of features not present in the feature subset under consideration), every time a feature is added, makes the system very time consuming and therefore computationally expensive classifiers like MLP and SVM cannot be used [38]. To account for this, instead of using a wrapper method, we propose a filter method to calculate heuristic desirability of the new subset.

Another very serious drawback of TFSACO and UFSACO is lack of a memory to store the best solutions. Without a memory the feature subsets generated in each iteration is lost. To counter this problem, here, we introduce a fitness-based memory. A fitness-based optimality allows us to save the best feature subsets with respect to accuracy and number of features (in each subset) present in each iteration. For quick reference, the drawbacks of the papers on FS using ACO and our modifications to counter the same are summarized in Table 1.

3.2 Proposed technique

For the sake of simplicity, hereafter, we use the alias WFACOFS (wrapper-filter ACO feature selection) for our proposed method. Flowchart of the entire work is given in Fig. 2.

Ants (denoted by G—contains features selected by the ant) moves from one feature to another feature, where probability of addition of a feature is calculated (using Eq. 5) which requires the calculation of two quantities—heuristic desirability (η_i) and pheromone density (ρ_i) for each feature.

As specified before, we use the concept of similarity as our filter method which is shown in Eq. 1 where the similarity between two features X and Y is calculated.

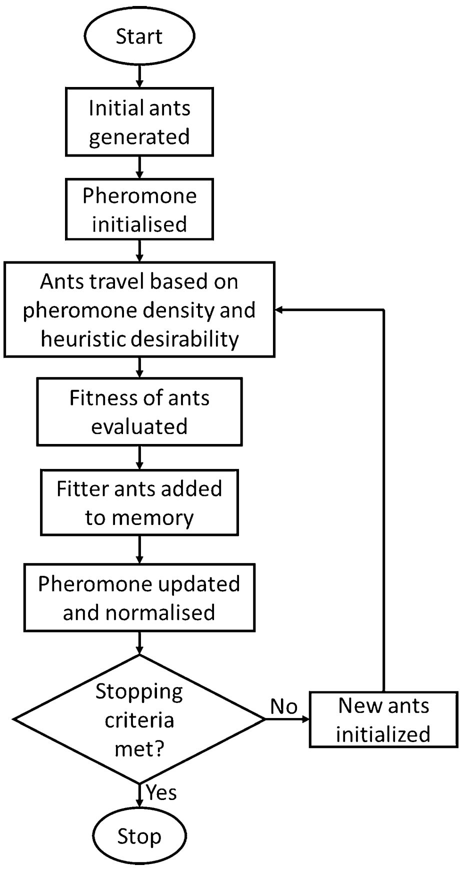
Start
Initial ants generated
Pheromone initialised
Ants travel based on pheromone density and heuristic desirability
Fitness of ants evaluated
Fitter ants added to memory
Pheromone updated and normalised
Stopping criteria met?
No
New ants initialized
Yes
Stop

Fig. 2 Flowchart for proposed WFACOFS

Table 1 Summarization of the previously used ACO-based methods along with the modifications we propose here to counter the same

Problem	TFSACO [29]	UFSACO [30]	Modification
Premature convergence	Global and local pheromone update rules used with absolute values. So, over the iterations certain pheromone values may become very high which results in increasing probability of selection of the feature to a large extent; thereby converging solution to a specific path	Global and local update rules used which depend only on the frequency of the FS. Therefore, it increases the selection of certain number of features again and again over the iterations	Update rule is similar but pheromone values are restricted in the range of [0-1] by dividing them with the maximum pheromone value. Hence, making pheromone values comparable and thus avoiding premature convergence and allowing exploration of search space
Population lost	The population information at a certain iteration is completely lost. So, their only contribution in constructing the solution is toward updating pheromone values and encouraging selection of certain features for next generation	Same as TFSACO	We have introduced a fitness-based memory to keep record of every generation of solutions. So, we can get the most suited ants among all generations
Approach	This method uses wrapper approach to generate the solution and test them at each step to modify the pheromone values	This method uses filter approach to generate solutions without testing them for quality	We have used a hybrid approach by combining both filter and wrapper methods. We have created the feature sets using filter method and using wrapper method we have evaluated their quality to update the pheromone values

$$\mathrm{sim}(X,Y)=\frac{\sum_{i=1}^{a}x_i*y_i}{\sqrt{\sum_{i=1}^{a}x_i^2}*\sqrt{\sum_{i=1}^{a}y_i^2}} \tag{1}$$

The use of similarity helps us to lower the computation cost by a huge extent. This lets us design a filter-wrapper combination for ACO.

η_i is estimated by calculating the value of similarity between the feature, last added, and the feature whose addition probability we are calculating. The value of similarity is 1 if the features are completely correlated and 0 if the features are completely independent. The objective is to include non-correlated features to increase the recognition capability of the system. The value of η_i is calculated using Eq. 2 where j denotes the feature last added and i denotes the feature we are trying to add. The value is bound between [0, 1].

$$\eta_i=\frac{1}{1+\mathrm{sim}(i,j)} \tag{2}$$

The values of the similarities are used multiple times and they are, therefore, computed offline so as to reduce the complexity and make the construction of the new ants a very cost-effective operation as opposed to the typical wrapper approaches.

ρ_i is initialized to zero and updated both locally and globally using Eq. 3. Local update value ($\Delta\rho_i^l$) is found using Eq. 4, while the global update is denoted by $\Delta\rho_i^g$. γ denotes the classifier used and the $\gamma(G)$ is the classification accuracy of the features selected by an ant. n is the number of features, t denotes the time, m the number of ants in the population and d is the pheromone evaporation factor.

$$\rho_i(t+1)=(1-d)\rho_i(t)+\sum_{l=1}^{m}\Delta\rho_i^l(t)+\Delta\rho_i^g(t) \tag{3}$$

$$\Delta\rho_i^l(t)=\begin{cases}\emptyset*\gamma(G)+\dfrac{(1-\emptyset)*(n-|G|)}{n}, & \text{if } i\in G\\ 0, & \text{otherwise}\end{cases} \tag{4}$$

The way pheromone is updated in Eq. 4 allows the success of the previous iterations to be reflected on the future generations. A balance needs to be struck between importance given to accuracy and the feature size, so we take the value of $\emptyset=0.8$, giving accuracy much more importance than accuracy. The global pheromone update is done by the best ant depositing pheromone twice. It is calculated according to Eq. 4, but only for the ant with the highest accuracy in that iteration (or generation).

The probability of addition of feature to an ant is calculated by Eq. 5. G is the set of features selected by the ant.

$$\text{Probability}=\frac{[\rho_i(t)]^{\alpha}*[\eta_i]^{\beta}}{\sum_{u\in G}[\rho_u(t)]^{\alpha}*[\eta_u]^{\beta}},\ \text{where } i\in G \tag{5}$$

The values of α,β provide us the necessary balance between the exploitation and exploration. The use of ρ_i (pheromone on the ith feature) provides us the scope of including previous success into decision making. So, if $\alpha=0$ then we perform FS without considering the history of previous successes. While if we make $\beta=0$ then the attractiveness or η_i of the move is neglected. Therefore, balance between the importance given to ρ_i and η_i provides an important aspect for setting a trade-off between exploitation and exploration of the search space. Considering this, the values of both α and β are taken as 1 and equal importance is given to both the aspects.

Pheromone serves as the goodness measure of a feature in classification of the pattern classes. Therefore, results of the past iterations help us to guide our search toward more productive feature subsets. While η_i, using similarity, allows us to minimize the redundancy in the feature subsets.

The features are selected by the ants following Eq. 5 as they travel. Ordering of features in which they are evaluated for selection by an ant plays an important role. Ordering of the features by pheromone value gives the features with a high pheromone value, a higher probability of being included every time. This leads to a self-predictive machine as the pheromones are deposited partially and therefore exploration ability of ACO is badly diminished. If it is included numerous times, the pheromone value increases dramatically leading to a lack of exploration of unvisited features. In our case, the features are randomly ordered for each ant to avoid any kind of biasness to a particular feature. The pheromone value is normalized by dividing this value with the maximum value to ensure the outcome lies between [0, 1].

Basic algorithm of ACO does not preserve the feature subsets derived previously nor does it, like elitism in GA, preserve the best of the feature subsets generated. So, to overcome this shortcoming, we add a fitness-based memory (of size $2*m$) to preserve the best feature subset in terms of accuracy and the number of features in the subset. The number of ants in the colony is m. Accuracy is given more priority than number of selected features by assigning a higher weight to the accuracy term (w_1) than the weight of the ratio of unselected features to the feature dimension (w_2). Equation 5 gives the objective function in determining the fitness of a feature subset—G, where n is the total number of features in the dataset. The function γ returns the classification accuracy of subset G.

$$\mathrm{fitness}(G)=w_1*\gamma(G)+w_2*\frac{(n-|G|)}{n} \tag{6}$$

The complexity of the method described in [29] is $O(Imn)$ where I is the number of iterations, m is the number of ants and n is the total number of features. In our case, as we can consider the calculation of probability of addition (calculation of heuristic desirability is of $O(1)$ complexity) to be of $O(1)$, so our complexity reduces to $O(Im)$ which means that for high dimensional feature set ($n \gg m$), where FS plays a very important role, our method would be far more effective. Our entire algorithm is given hereafter.

WFACOFS Algorithm

Calculate Similarity matrix (offline) using equation 1 *and* 2

Inputs:

m: Number of ants in colony

iteration: Number of generations

d: Pheromone evaporation factor

α: Relative importance to pheromone value

β: Relative importance to heuristic desirability (similarity value)

Output: The best set of ants according to the proposed method

```
1:   Begin
2:   Create initial population of m ants, each of dimension $(1, n)$ where
         n is the total number of features present in the dataset
3:   for $i = 1$ to iteration do
4:       for $j = 1$ to $m$ do
5:           Create new ants with empty feature set
6:           Randomly order all the features for ant j
7:           for $k = 1$ to $n$ do
8:               Calculate probability of addition of the kth
                     feature for ant j using equation 5
9:               if (probability > some random value)
10:                  Select the feature and move the jth ant
                         to the newly selected feature
11:              end if
12:          end for
13:      end for
14:      Find the fitness of the ants and
             rank them according to their accuracy
15:      Update memory with the newly created ants
16:      Update pheromone values considering deposition by
             each ant using equations 3 and 4
17:  end for
18:  End
```

So, to sum it up, our contributions in this paper are

(1) A combination of filter and wrapper methods is used to generate the solution (feature subset) with far less computation cost.

(2) Introduction of memory, ants to keep track of the best solution generated throughout all generations.

(3) An exhaustive study on real-life datasets to show the utility of this approach.

4 Experimental results

Theoretically, the proposed filter-wrapper version of ACO has been shown to have a lower time complexity than wrapper version of ACO. The introduction of the memory allows us to retain the best results. This section presents the results obtained by the proposed method and its comparison with several other well-known approaches. The experimentation is done on some popular datasets chosen from the UCI repository and NIPS2003 FS challenge. All the experimentations of this approach are implemented in MATLAB and tested on an Intel Core-i3 CPU with 4 GB of RAM. The corresponding datasets and their information are listed in Table 2.

4.1 Datasets

For the proper evaluation of our proposed algorithm, we have selected different types of datasets from various fields

Table 2 Description of the datasets used in the present work

Dataset	Symbol	Number of samples	Number of classes	Number of features
Wine	WI	178	3	13
Soybean-small	SS	35	4	47
Ionosphere	IO	351	2	34
Breast Cancer	BC	699	2	9
Monk2	MK2	169	2	6
Hill-Valley	HV	606	2	101
Monk1	MK1	124	2	6
Arrhythmia	ARR	452	16	279
Horse	HR	368	2	27
Madelon	MN	4400	2	500

which can be clearly differentiated with respect to the number of features and can be categorized as

(a) Small (Number of features < 10)

- Monk1
- Monk2
- Breast Cancer

(b) Medium ($10 \leq$ Number of features ≤ 100)

- Wine
- Horse
- Ionosphere
- Soybean-small

(c) Large (Number of features > 100)

- Arrhythmia
- Hill-valley
- Madelon

Table 3 Parameter description along with their corresponding values as used in our implementation

Parameter symbol	Parameter significance	Value
n	Number of ants	10
α	Exploitation balance factor	1
β	Exploration balance factor	1
Iteration	Number of iterations	20
d	Pheromone evaporation factor	0.15
$\emptyset$	Pheromone evaluation factor	0.8
w_1	Weight of accuracy	100
w_2	Weight of number of features	1

4.2 Classifiers used

After obtaining the feature subset through the filter method, we have used classifiers to evaluate the final solutions

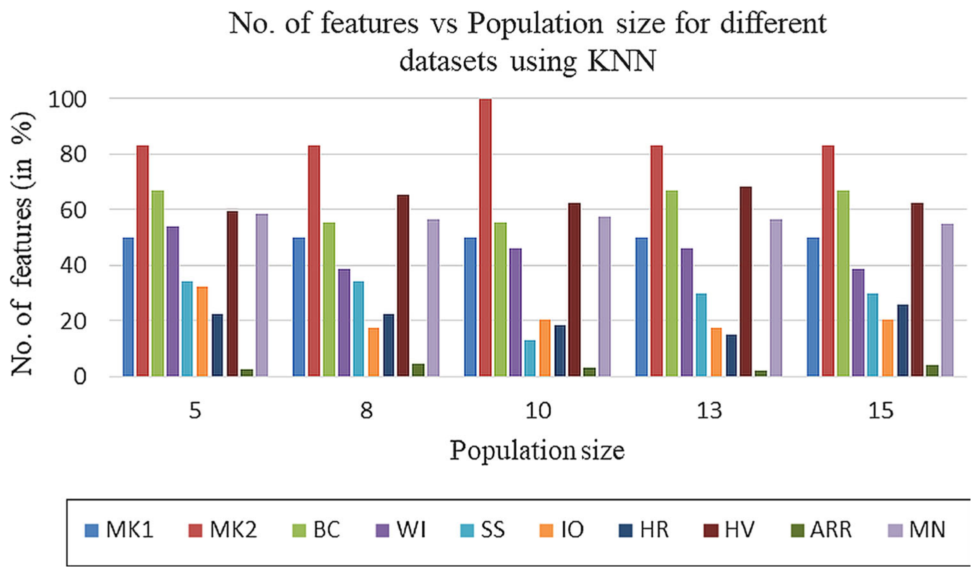

Fig. 3 Number of features versus population size for all 10 datasets using KNN classifier

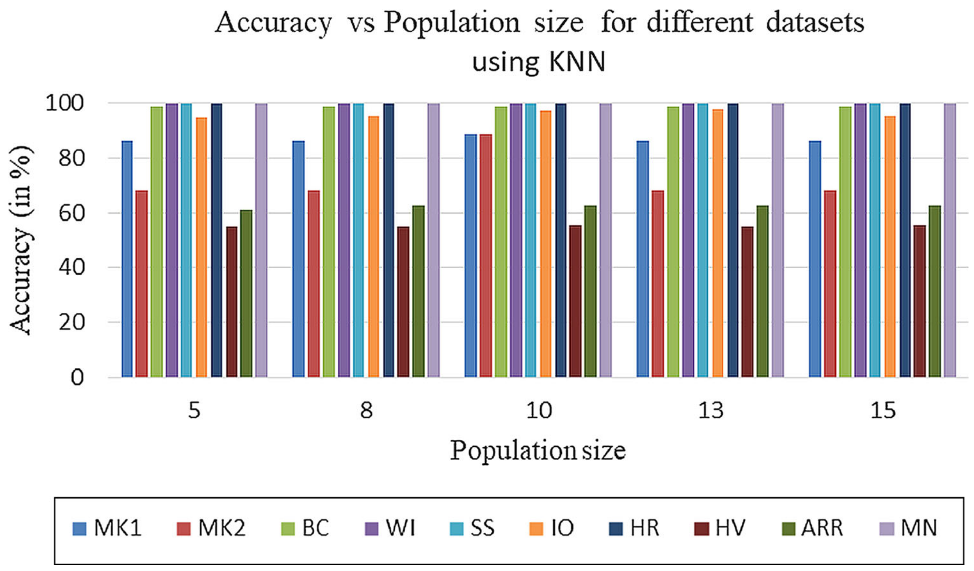

Fig. 4 Accuracy versus population size for all 10 datasets using KNN classifier

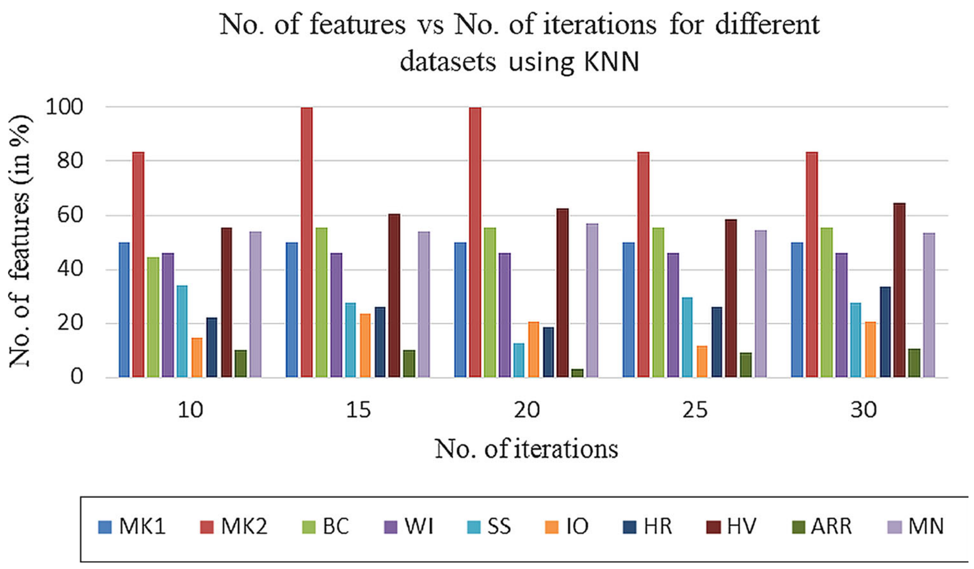

Fig. 5 Number of features versus number of iterations for all 10 datasets using KNN classifier

obtained by the ants in each iteration. The classifiers which are used in the proposed method are K-Nearest Neighbors (KNN) and MLP.

KNN is a well-established algorithm to classify an object with respect to its nearest neighbors. The output is a class membership which is obtained by taking votes from its K-nearest neighbors. This classifier is very popular due to its simplicity and efficiency at the same time. On the other hand, MLP is a popularly used and efficient classifier. It is a feed-forward artificial neural network which consists of three set of layers—input, hidden and output layers. The layers form a connected graph and are assigned random weights which are modified during training using back-propagation algorithm. So, here we have used both computationally inexpensive (KNN) and expensive classifiers (MLP) to evaluate our wrapper-filter-based approach. This is possible due to lower complexity of our model.

4.3 Parameter setting

In the proposed method, we have used several parameters which are adjusted to suit the purpose of the approach. Though already mentioned, but for easy reference, the values of the parameters which we have used are recorded in Table 3.

We have also tested our method by varying the population size and the number of iterations and the results of the same are plotted in Figs. 3, 4, 5, 6, 7, 8, 9 and 10. The number of ants in the population is varied, for both MLP and KNN classifiers. Accuracy variations are shown in

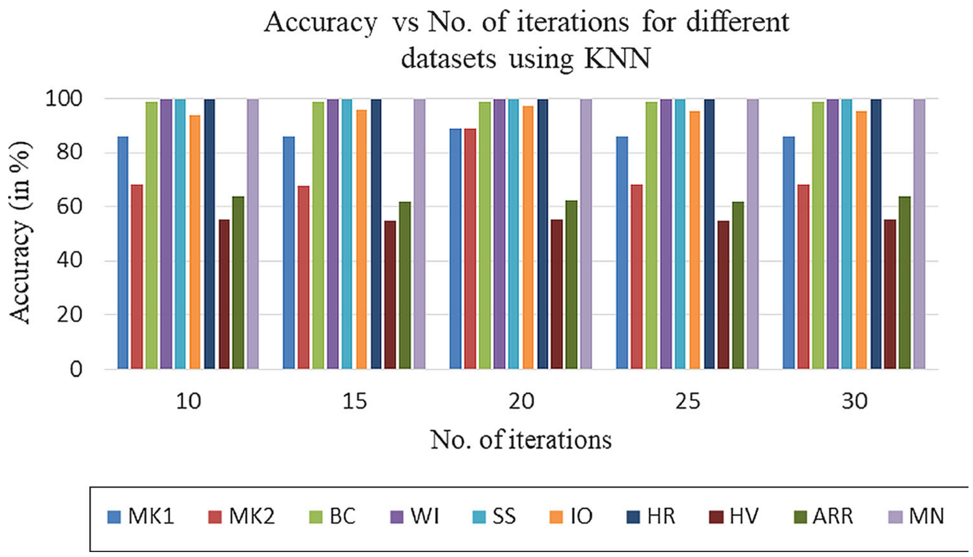

Fig. 6 Accuracy versus number of iterations for 10 datasets using KNN classifier

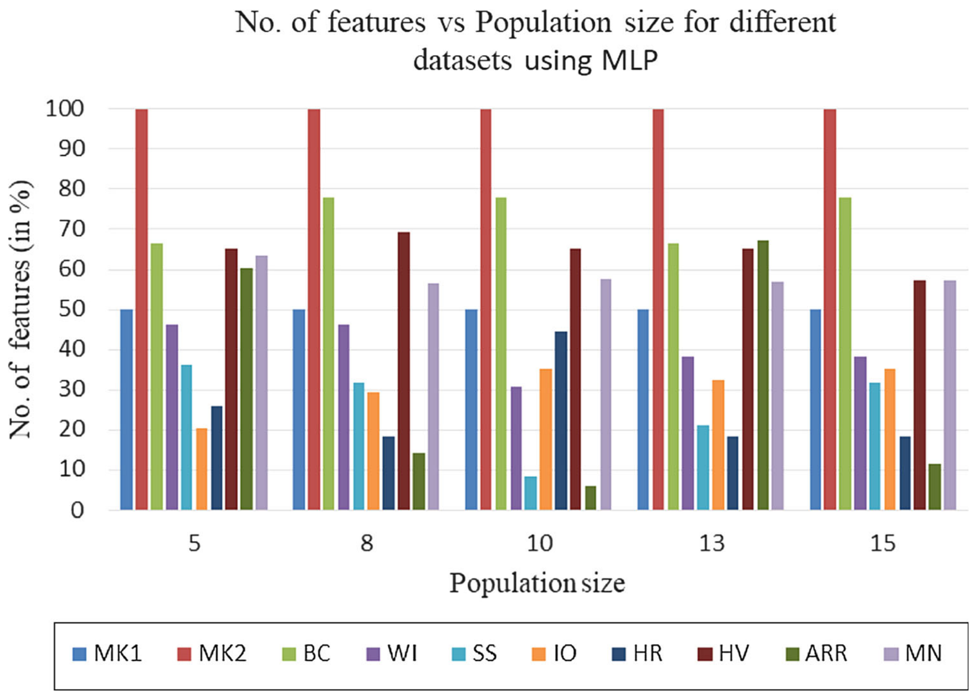

Fig. 7 Number of features versus population size for all 10 datasets using MLP classifier

Figs. 4 and 8 and selected feature dimension variations are given in Figs. 3 and 7. The number of iterations is varied for MLP and KNN classifiers. Figures 5 and 9 show the variation in the selected feature dimension, and Figs. 6 and 10 depict variations in accuracy.

4.4 Results and comparison

First, the performance of WFACOFS is evaluated for different classifiers. Then we have made comparison with other ACO algorithms proposed recently as well as with some previously proposed versions of ACO algorithms and other FS algorithms namely BCOFS, HMOGA, ME-PSO and SA. List of the ACO variants which we have used for comparison are provided in Table 4. All the related information and comparison results are enlisted in Tables 5 and 6. In most of the cases, WFACOFS has obtained an edge over other ACO algorithms and the other FS algorithms.

Table 5 lists the results of our method on ten datasets using both KNN and MLP. Except MK1, HR and MN, rest of the seven datasets give better accuracy using MLP. The

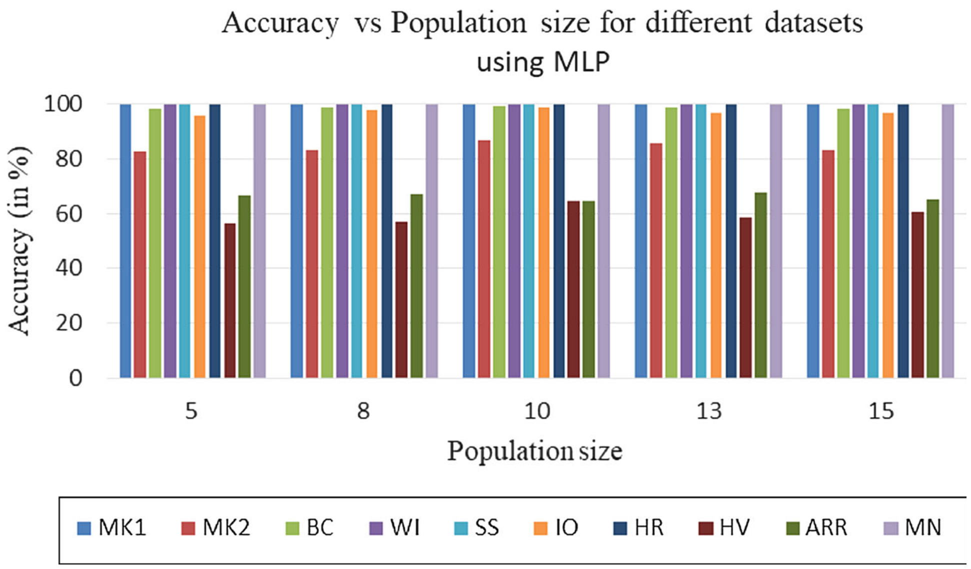

Fig. 8 Accuracy versus population size for all 10 datasets using MLP classifier

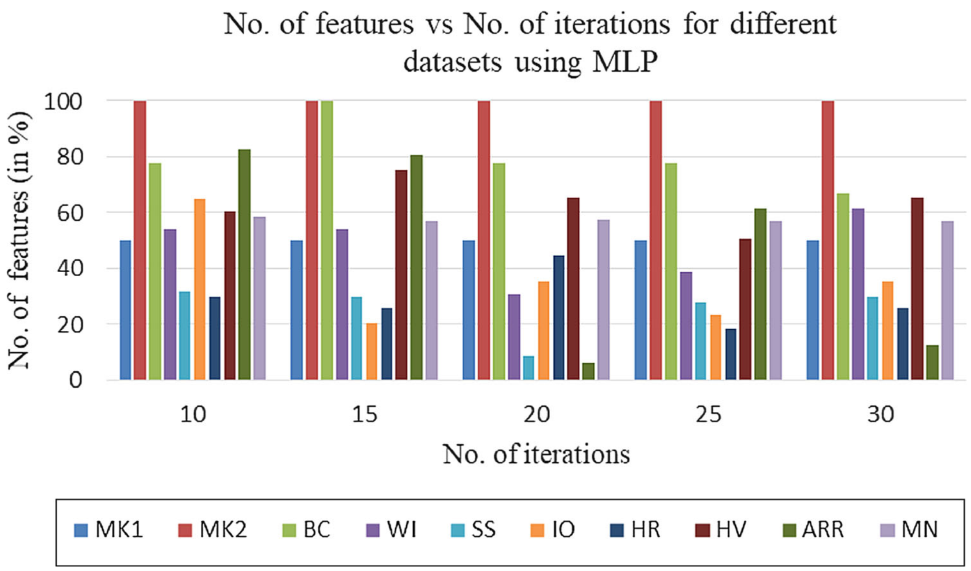

Fig. 9 Number of features versus number of iterations for all 10 datasets using MLP classifier

average classification accuracy and standard deviation for 10 runs of WFACOFS are also provided in Table 5.

From Table 6, we can observe that for eight out of 10 datasets, WFACOFS has achieved the best accuracy and for the rest, it has achieved second best accuracy. The list of datasets over which WFACOFS has achieved best results are MK1, MK2, WI, IO, HV, HR and ARR. Even though in case of ARR, accuracy of classification given by WFACOFS is lower than I-RACOFS, we have considered the result given by our proposed algorithm as better because it has been able to decrease the number of features to a large extent. For ARR, the difference in their accuracies is only 0.37% but number of features selected by WFACOFS is only 17 which is very low in comparison with I-RACOFS which is 56. That is why we may state that WFACOFS gives a better result. Over rest of the datasets, i.e., BC and SS, we have reached the second best result but the difference is negligible. In case of SS, we have reached 100% accuracy but the only difference is that WFACOFS has taken a greater number of features in comparison with RACOFS and C-RACOFS. For BC, the difference between the best result and WFACOFS result is only 0.13% in terms of accuracy. In a nutshell, we can mention that no single method performs better than WFACOFS in more than one dataset. For the datasets in which we are outperformed, it is only by a very small margin. In MN we get 100% accuracy but SA performs better feature reduction than us, though by

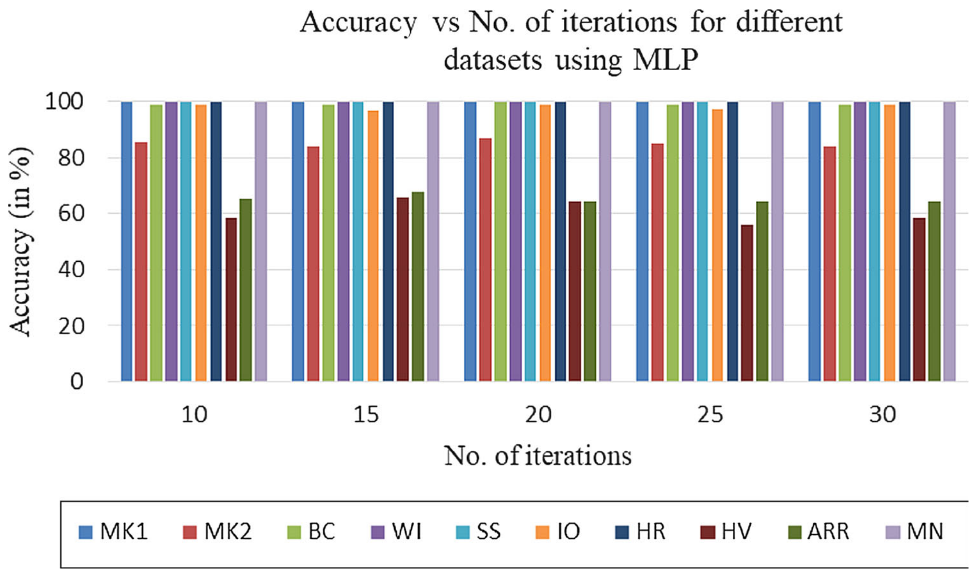

Fig. 10 Accuracy versus number of iterations for all 10 datasets using MLP classifier

Table 4 State-of-the-art FS algorithms used for comparison with the proposed method

Different FS algorithms with reference	Alias used
Unsupervised FS ACO [12]	UFSACO
enRiched ant colony optimization [6]	RACOFS
Capability RACOFS [8]	C-RACOFS
Improver RACOFS [8]	I-RACOFS
ACO feature selection-P [39]	ACOFS-P
ACO feature selection-R [39]	ACOFS-R
Two-Stage updating pheromone for invariant ACO 1 [28]	TSI ACO1
Two-Stage updating pheromone for invariant ACO 2 [28]	TSI ACO2
Ant system [25]	AS
Max–min ant system [27]	MMAS
Ant colony system [26]	ACS
Histogram-based genetic algorithm [23]	HMOGA
Bee colony optimization feature selection [14]	BCOFS
Text feature selection ACO [29]	TFSACO
Mutation-enhanced particle swarm optimization [22]	ME-PSO
Simulated annealing [11]	SA

a small margin (SA performs 6% more feature reduction than WFACOFS).

From Table 6, we can also conclude that WFACOFS gives impressive results for medium- and small-sized datasets. These results are superior to all the other ACO versions in terms of accuracy as well as other FS algorithms like BCOFS, ME-PSO, HMOGA and SA. For large datasets, its performance is a bit low but it is comparable to other ACO algorithms. An interesting note in this case is that in case of large datasets our method can reduce the feature dimension by a significant margin compared to its contemporaries.

4.5 Time analysis

In this section, we analyze the time requirement for our proposed algorithm over different types of datasets with varying parameters such as number of ants and number of iterations. Although WFACOFS requires more time than filter approaches, it compensates for the time requirement with a significant increase in classification accuracy (comparison in Table 6 between UFSACO and WFACOFS). So, an efficient time-accuracy trade-off is achieved in WFACOFS. Theoretically, our ACO has a lower time complexity than all its contemporary wrapper-based

Table 5 Result of the proposed WFACOFS for different classifiers

Category	Dataset	Classifier KNN						MLP					
		Best		Average		Standard deviation		Best		Average		Standard deviation	
		Number of features	Accuracy (%)	Number of features	Accuracy (%)	Number of features	Accuracy (%)	Number of features	Accuracy (%)	Number of features	Accuracy (%)	Number of features	Accuracy (%)
	MK1	3	88.89	3.0000	86.1111	0.0000	0.0000	**3**	**100**	99.9074	3.1000	0.2928	0.3162
Small	MK2	**6**	**88.89**	5.0000	68.2870	0.0000	0.0000	6	87.04	83.2176	6.0000	1.7739	0.0000
	BC	5	99	5.6000	98.9967	0.0000	0.0000	**7**	**99.67**	99.1973	5.7000	0.1727	0.9487
	WI	6	100	6.1000	100.0000	0.0000	0.0000	**4**	**100**	100	6.1000	0	0.7379
	IO	7	97.35	7.2000	96.1589	0.0081	0.0081	**12**	**98.68**	97.6159	12.5000	0.342	6.2227
Medium	SS	6	100	13.4000	100.0000	0.0000	0.0000	**4**	**100**	100	13.7000	0	0.9487
	HR	**5**	**100**	6.2000	100.0000	0.0000	0.0000	12	100	100	5.8000	0	1.0328
	HV	63	55.61	61.4000	55.4212	0.0022	0.0022	**66**	**64.52**	57.674	65.9000	1.3163	5.4863
Large	ARR	9	62.5	12.7000	62.4342	0.0130	0.0130	**17**	**64.47**	67.3684	219.7000	1.4938	6.508
	MN	**286**	**100**	325.4000	99.4667	0.5670	0.5670	288	100	99.67	315.3000	0.9921	2.3145

The best results obtained are made bold

methods. To prove the time effectiveness of our method experimentally, we have compared the time requirements of WFACOFS with a well-known wrapper method TFSACO over three different types of datasets—one small (MK1), a medium (WI) and one large (HV). We have varied the number of ants and the number of iterations to compare these two methods in different conditions. The graphical comparison results are plotted in Figs. 11, 12, 13, 14, 15 and 16. As evident, the proposed algorithm requires far less computation time in comparison with TFSACO. Another interesting feature is that our algorithm's computation cost is far less sharp than TFSACO which proves the scalability of our algorithm.

4.6 Evaluation of robustness

Apart from the said UCI datasets, we have performed some additional testing of WFASCOFS to prove its robustness. The model is therefore applied on Facial Emotion Recognition (FER) dataset and cancer classification (here microarray) datasets. For FER, RaFD [45] dataset has been used in this experimentation. It contains pictures of people (male and female of varying ages) with eight different facial expressions namely anger, disgust, fear, neutral, sadness, surprise and contempt. The pictures containing only the frontal gaze are taken. There are 201 images in each class. Each image is resized to three sizes—32 $\times$ 32, 48 $\times$ 48 and 64 $\times$ 64. This allows for accounting for model performance due to variation in image dimension. The detail results are provided in Table 7. It is to be noted that from the RaFD dataset Gabor filter descriptor [46] is used to extract features on which FS is done.

Cancer classification on microarray datasets is performed using the datasets reported in [2]. They have used seven microarray datasets. The forms of cancer dealt with are Myeloid Leukemia (AMLGSE2191), Colon cancer, DLBCL, Leukemia, Prostate, Mixed-lineage Leukemia (MLL) and small round blue cell tumors (SRBCT). The details of the microarray datasets used for evaluation of WFACOFS are provided in Table 8.

In both the cases, our proposed model has achieved significant improvement in accuracy over the situation when entire feature set is used for classification. The classification accuracies obtained without FS and after performing FS with WFACOFS are provided in Tables 9 and 10 for FER and microarray datasets, respectively. For FER, SVM classifier is used with linear kernel and Sequential Minimal Optimization as the solver. For microarray datasets, MLP classifier is used with 20 neurons in the hidden layer.

Tables 9 and 10 contain results for FER and microarray data, respectively. From Tables 9 and 10, we can clearly see that WFACOFS is able to successfully reduce the

Table 6 (a, b) Comparison of our proposed approach with 16 other well-established FS algorithms

Dataset	RACOFS	C-RACOFS	I-RACOFS	TFSACO	UFSACO	ACOFS-P	ACOFS-R	TSI ACO1	WFACOFS
(a) Comparison of WFACOFS with eight contemporary methods									
MK1	**100(3)**	**100(3)**	97.2(4)	82.87(5)	72.22(3)	**100(3)**	**100(3)**	**100(3)**	**100(3)**
MK2	79.4(6)	79.4(6)	68.7(5)	64.35(3)	62.04(3)	68.3(3)	68.3(3)	67.12(2)	**88.89(6)**
BC	96.9(2)	96.1(4)	**99.8(2)**	98.33(6)	91.64(6)	94.7(3)	94.7(3)	96.85(4)	99.67(7)
WI	79.21(6)	79.21(6)	79.21(5)	95.74(8)	95.08(5)	66.4(5)	88.7(3)	94.94(6)	**100(4)**
IO	94.01(15)	94.01(15)	95.15(7)	96.03(22)	88.61(30)	57.4(12)	90.88(6)	91.16(14)	**98.68(12)**
SS	**100(2)**	**100(2)**	100(7)	100(21)	100(19)	100(3)	100(14)	100(9)	100(4)
HR	80.8(3)	80.8(3)	88.3(6)	72.06(17)	55.85(15)	59.7(4)	60(3)	72.4(10)	**100(5)**
HV	35.4(10)	48.7(52)	60.06(60)	52.56(68)	60.44(55)	58.5(48)	59.6(36)	50.62(18)	**64.52(66)**
ARR	55.7(273)	56.1(275)	64.85(56)	56.58(166)	59.22(20)	56.51(138)	56.43(130)	57.74(241)	**64.47(17)**
MN	88.39(241)	89.96(312)	92.29(199)	88.13(346)	61.06(70)	81.04(147)	88.04(376)	89.54(384)	100(286)
AVG	80.16(36)	81.59(41)	83.7(17)	79.81(35)	76.12(18)	73.50(25)	79.84(22)	81.20(34)	**90.69(14)**
Dataset	TSI ACO2	AS	MMA	ACS	HMOGA	MPSO	BCOFS	SA	WFACOFS
(b) Comparison of WFACOFS with eight more contemporary methods									
MK1	**100(3)**	68.45(3)	**100(3)**	**100(3)**	83.33(3)	88.89(3)	50(2)	72.22(3)	**100(3)**
MK2	67.12(2)	64.26(3)	64.26(3)	64.26(3)	55.09(2)	74.7685(6)	67.17(4)	56.02(3)	**88.89(6)**
BC	95.7(3)	94(2)	94(2)	92.4(1)	96.32(3)	98.9967(5)	85.4(5)	96.99(5)	99.67(7)
WI	95.5(8)	91.6(11)	92.15(7)	86.3(6)	70.21(5)	**100(4)**	95(8)	97.87(7)	**100(4)**
IO	91.16(14)	88.9(10)	85(12)	91(13)	93.38(17)	96.02(13)	93.16(15)	92.05(18)	**98.68(12)**
SS	100(10)	100(7)	100(7)	100(7)	85.71(19)	100(11)	100(5)	92.86(13)	100(4)
HR	70.6(12)	74.8(6)	77.15(6)	81.3(6)	97.05(14)	100(7)	65.23(13)	61.76(18)	**100(5)**
HV	52.4(40)	61.45(29)	60.46(39)	59.14(58)	51.50(54)	53.66(37)	54.6(62)	51.65(42)	**64.52(66)**
ARR	56.1(275)	60.95(28)	62.27(22)	60.62(23)	56.58(122)	63.15(144)	56.2(139)	58.55(144)	**64.47(17)**
MN	89.26(328)	86.30(287)	90.08(143)	90.05(106)	60.33(240)	100(280)	81.66(169)	**100(256)**	100(286)
AVG	80.95(40)	78.27(11)	81.70(11)	81.67(13)	74.85(23.9)	87.55(50.9)	74.06(28)	78.43(50.9)	**90.69(14)**

The number of features corresponding to each experiment is provided in brackets. The best results are marked in bold

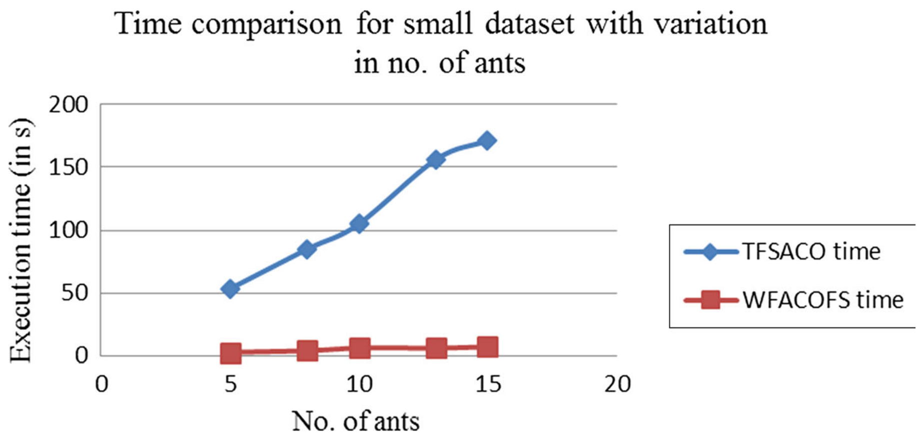

Fig. 11 Execution time versus number of ants for TFSACO and WFAOFS over a small dataset

feature dimension of the datasets as well as increase the classification accuracy significantly. In case of microarray datasets, our proposed model is able to achieve 100% accuracy in five out of six datasets with an impressive reduction in feature dimension. For FER dataset, WFACOFS has achieved a significant improvement in

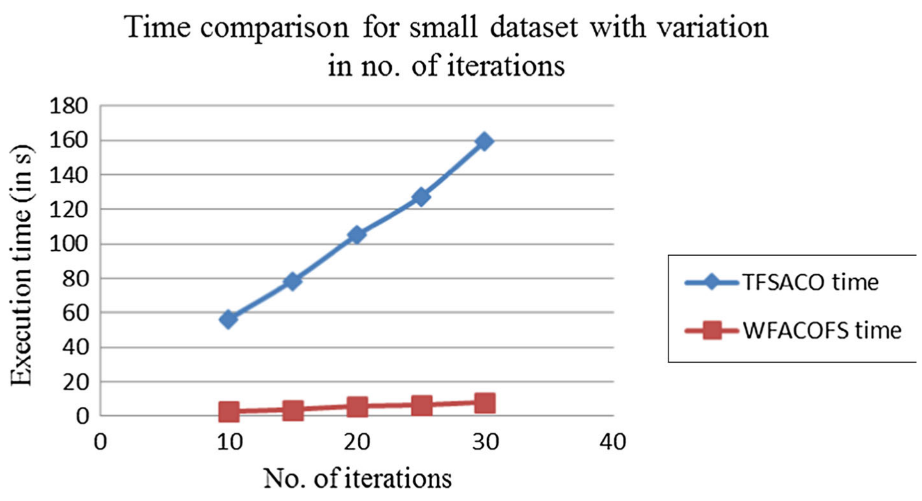

Fig. 12 Execution time versus number of iterations for TFSACO and WFAOFS over a small dataset

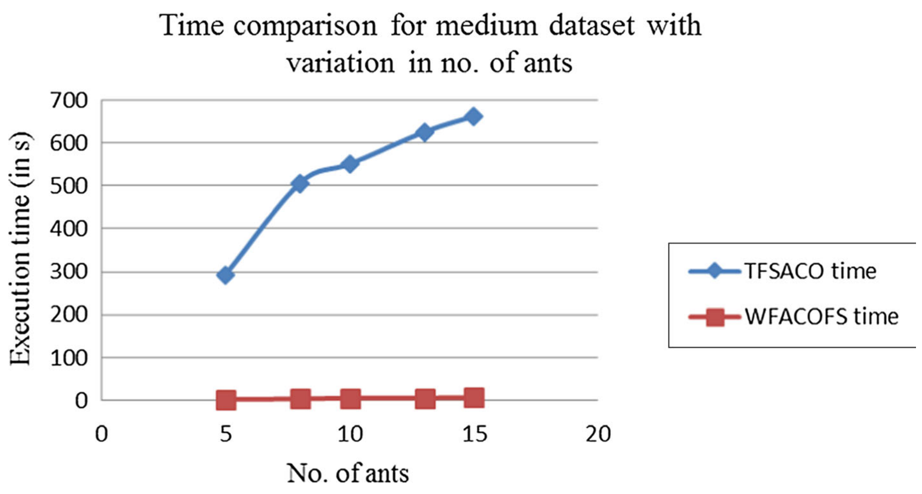

Fig. 13 Execution time versus number of ants for TFSACO and WFAOFS over a medium dataset

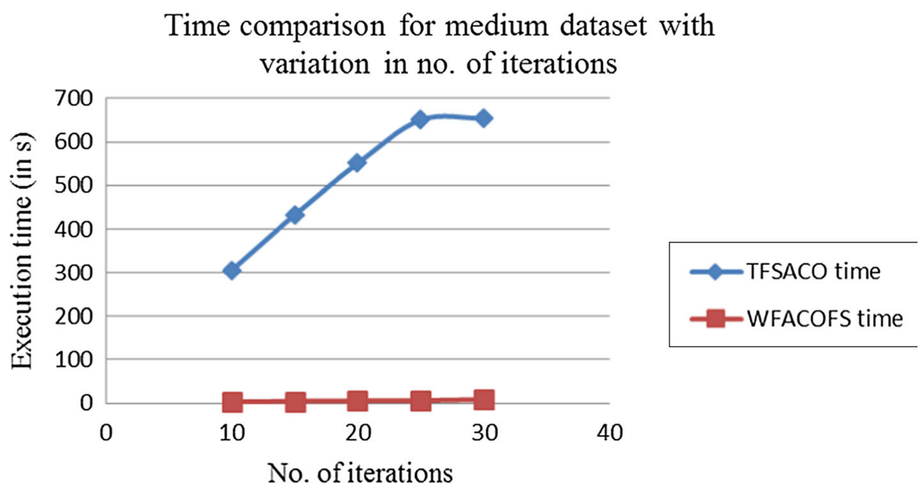

Fig. 14 Execution time versus number of iterations for TFSACO and WFAOFS over a medium dataset

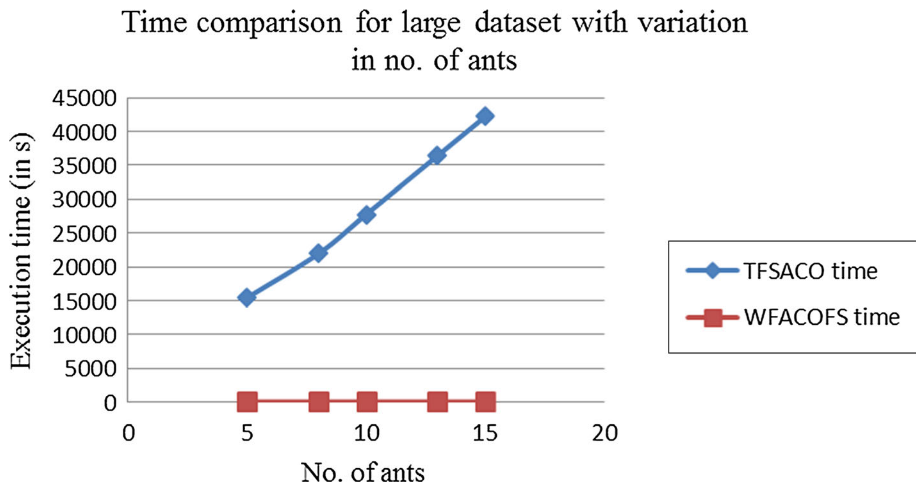

Fig. 15 Execution time versus number of ants for TFSACO and WFAOFS over a large dataset

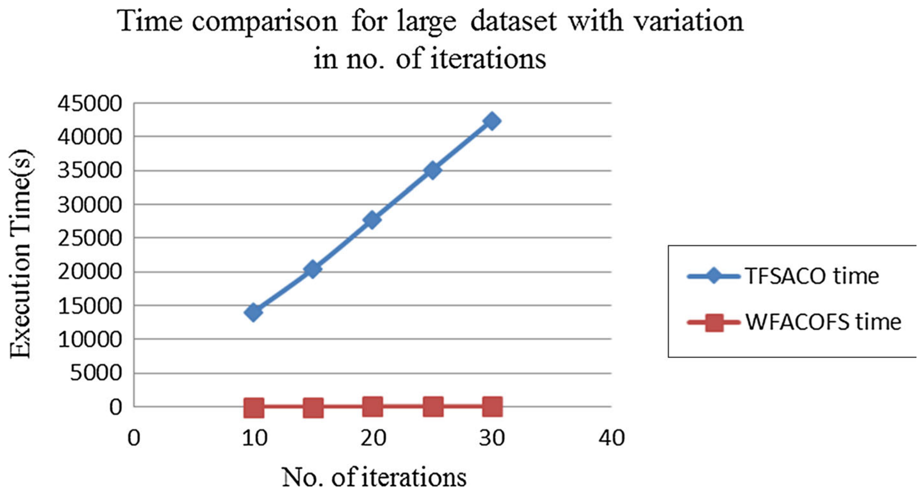

Fig. 16 Execution time versus number of iterations for TFSACO and WFAOFS over a large dataset

Table 7 Description of FER datasets (Gabor filter-based features) used to evaluate the WFASCOFS model

Dataset	Image dimension	Number of features	Number of samples	Number of classes
RaFD	**32 × 32**	640	1608	8
	48 × 48	1440		
	64 × 64	2560		

Table 8 Description of microarray datasets used to evaluate the WFASCOFS model

Dataset	Number of features	Number of samples	Number of classes
AMLGSE2191	12,616	54	2
Colon	7464	36	2
DLBCL	7070	77	2
Leukemia	5147	72	2
Prostate	12,533	102	2
MLL	12,533	72	3
SRBCT	2308	83	4

Table 9 Results obtained by WFACOFS on FER dataset using SVM classifier

Dataset	Image dimension	Accuracy on whole feature vector (in %)	Original feature dimension	WFACOFS results	
				Accuracy (in %)	Number of selected features
RaFD	**32 × 32**	82.56	640	91.6	456
	48 × 48	88.53	1440	97.2	1154
	64 × 64	85.83	2560	98.5	2134

Table 10 Results obtained by WFACOFS on microarray datasets using MLP classifier

Microarray dataset	Accuracy on whole dataset (%)	Original feature dimension	WFACOFS results	
			Accuracy (in %)	Number of selected features
AMLGSE2191	51.85	12,616	96.3	17
Colon	88.89	7464	100	3
DLBCL	76.92	7070	100	3
Leukemia	83.78	5147	100	5
Prostate	62.75	12,533	100	22
MLL	68.57	12,533	100	25
SRBCT	85	2308	100	19

Table 11 Comparison of classification accuracy obtained by WFACOFS with other well-established FS methods on FER dataset

Dataset	Image size	Accuracy (in %)					
		GA	MA	SA	TFSACOFS	I-RACOFS	WFACOFS
RAFD	**32 × 32**	93.28(400)	94.03(429)	83.21(320)	89.05(488)	90.4(373)	91.6(456)
	48 × 48	98.32(851)	98.88(894)	91.6(683)	93.2(1023)	95.7(864)	97.2(1154)
	64 × 64	98.32(1613)	98.75(1414)	95.9(1333)	97.94(2368)	97.3(1572)	98.5(2134)

The number of features corresponding to each experiment is provided in brackets

Table 12 Comparison of classification accuracy obtained by WFACOFS with other well-established FS methods on microarray datasets

Dataset	Accuracy (in %)					
	GA	MA	RMA	TFSACOFS	I-RACOFS	WFACOFS
AMLGSE2191	100(98)	100(91)	100(6)	88.18(28)	98.2(16)	96.3(17)
Colon	100(81)	100(81)	100(2)	96.4(26)	100(11)	100(3)
DLBCL	100(88)	100(105)	100(3)	85.05(24)	99.4(22)	100(3)
Leukemia	100(85)	100(65)	100(4)	96.59(31)	100(17)	100(5)
Prostate	100(99)	100(107)	100(3)	88.31(31)	98.7(18)	100(22)
MLL	100(94)	100(80)	100(4)	97.29(32)	100(13)	100(25)
SRBCT	100(78)	100(50)	100(5)	100(25)	100(17)	100(19)

The number of features corresponding to each experiment is provided in brackets

classification accuracy while using only 70–80% of the original features. The obtained results clearly prove the robustness of the proposed model to different FS problems. To check for the goodness of the results, we have further compared the obtained accuracy with the accuracies achieved by some popular FS algorithms. For microarray datasets, we have compared the results with GA, memetic algorithm (MA), recursive memetic algorithm (RMA) and two versions of ACO namely TFSACOFS, I-RACOFS. On the other hand, in case of FER dataset, we have used GA, MA, SA, TFSACOFS and I-RACOFS for comparison. Tables 11 and 12 record the comparison results for FER and microarray datasets, respectively.

From Tables 11 and 12, it can be observed that the proposed model is comparable to all the popularly used FS techniques applied here for comparison. WFACOFS even performs comparably to RMA which is a recently proposed technique designed specifically to perform FS in microarray data. Thus, we can safely state that WFACOFS is a model which is applicable to FS problem of any domain. It uses the power of a wrapper approach to enhance the classification ability and a filter approach to reduce the computational cost of the system which makes it an overall robust embedded model.

5 Conclusions and future work

Use of a filter method alongside a wrapper method to design an embedded system is a well-established trend and as evident from our work which brings fruitful results. In this work, we propose a new idea for calculation of heuristic desirability in ACO making the algorithm computationally efficient than its ancestors. Introduction of fitness-based memory and updating pheromone quantity using both accuracy and number of features allow us to perform FS in a multi-objective manner. Therefore, our algorithm can be considered as a computationally inexpensive multi-objective filter-wrapper method. Successful application on UCI datasets of varying sizes proves the effectiveness of our method. The proposed method has been further tested on microarray and FER datasets to evaluate its robustness. An interesting future scope would be exploration of the new ways to measure heuristic desirability using other filter methods in place of similarity. The deposit of pheromones on features instead of paths is an important part of our idea which can be further extended by fixing an adaptive ordering of features. For evaluating the fitness values, we can also apply some other classifiers like SVM and ELM.

References

1. Contributors W (2015) Curse of dimensionality. Wikipedia, Free Encycl
2. Ghosh M, Begum S, Sarkar R et al (2019) Recursive Memetic Algorithm for gene selection in microarray data. Expert Syst Appl 116:172–185. https://doi.org/10.1016/j.eswa.2018.06.057
3. Liu H, Motoda H (2007) Computational methods of feature selection. CRC Press, Boca Raton
4. Mitra P, Murthy CA, Pal SK (2002) Unsupervised feature selection using feature similarity. IEEE Trans Pattern Anal Mach Intell 24:301–312
5. Shang W, Huang H, Zhu H et al (2007) A novel feature selection algorithm for text categorization. Expert Syst Appl 33:1–5
6. Yang J, Honavar V (1998) Feature subset selection using a genetic algorithm. IEEE Intell Syst Appl 13:44–49
7. Moradi P, Gholampour M (2016) A hybrid particle swarm optimization for feature subset selection by integrating a novel local search strategy. Appl Soft Comput J 43:117–130. https://doi.org/10.1016/j.asoc.2016.01.044
8. Forsati R, Moayedikia A, Jensen R et al (2014) Enriched ant colony optimization and its application in feature selection. Neurocomputing 142:354–371. https://doi.org/10.1016/j.neucom.2014.03.053
9. Duval B, Hao J-K, Hernandez Hernandez JC (2009) A memetic algorithm for gene selection and molecular classification of cancer. In: Proceedings of 11th annual conference genetic evolutionary computation—GECCO'09 201. https://doi.org/10.1145/1569901.1569930
10. Zhu Z, Ong YS, Dash M (2007) Markov blanket-embedded genetic algorithm for gene selection. Pattern Recognit 40:3236–3248. https://doi.org/10.1016/j.patcog.2007.02.007
11. Fogel DB (1994) An introduction to simulated evolutionary optimization. IEEE Trans Neural Netw 5:3–14
12. Gandomi AH, Yang X-S, Alavi AH (2013) Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Eng Comput 29:17–35
13. Karaboga D, Akay B (2009) A comparative study of artificial bee colony algorithm. Appl Math Comput 214:108–132
14. Forsati R, Moayedikia A, Keikha A, Shamsfard M (2012) A novel approach for feature selection based on the bee colony optimization. Int J Comput Appl 43:30–34
15. Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67
16. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61
17. Jain M, Singh V, Rani A (2019) A novel nature-inspired algorithm for optimization: squirrel search algorithm. Swarm Evol Comput 44:148–175
18. Li MD, Zhao H, Weng XW, Han T (2016) A novel nature-inspired algorithm for optimization: virus colony search. Adv Eng Softw 92:65–88
19. Mafarja M, Mirjalili S (2018) Whale optimization approaches for wrapper feature selection. Appl Soft Comput 62:441–453
20. Mafarja MM, Mirjalili S (2017) Hybrid whale optimization algorithm with simulated annealing for feature selection. Neurocomputing 260:302–312
21. Emary E, Zawbaa HM, Hassanien AE (2016) Binary grey wolf optimization approaches for feature selection. Neurocomputing 172:371–381. https://doi.org/10.1016/j.neucom.2015.06.083
22. Wei J, Zhang R, Yu Z et al (2017) A BPSO-SVM algorithm based on memory renewal and enhanced mutation mechanisms for feature selection. Appl Soft Comput J 58:176–192. https://doi.org/10.1016/j.asoc.2017.04.061
23. Ghosh M, Guha R, Mondal R et al (2018) Feature selection using histogram-based multi-objective GA for handwritten Devanagari numeral recognition. In: Bhateja V, Coello Coello C, Satapathy S, Pattnaik P (eds) Intelligent engineering informatics. Advances in intelligent systems and computing. Springer, Singapore, pp 471–479. https://doi.org/10.1007/978-981-10-7566-7_46
24. Dorigo M, Stützle T (2019) Ant colony optimization: overview and recent advances. In: Gendreau M, Potvin JY (eds) Handbook of metaheuristics. Springer, Cham, pp 311–351. https://doi.org/10.1007/978-3-319-91086-4_10

25. Dorigo M, Maniezzo V, Colorni A (1996) Ant system: optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybern Part B 26:29–41
26. Gambardella LM, Dorigo M (1996) Solving symmetric and asymmetric TSPs by ant colonies. In: Proceedings of IEEE international conference on Evolutionary computation, 1996. IEEE, pp 622–627
27. Stützle T, Hoos HH (2000) MAX–MIN ant system. Futur Gener Comput Syst 16:889–914
28. Zhang Z, Feng Z (2012) Two-stage updating pheromone for invariant ant colony optimization algorithm. Expert Syst Appl 39:706–712
29. Aghdam MH, Ghasem-Aghaee N, Basiri ME (2009) Text feature selection using ant colony optimization. Expert Syst Appl 36:6843–6853. https://doi.org/10.1016/j.eswa.2008.08.022
30. Tabakhi S, Moradi P, Akhlaghian F (2014) An unsupervised feature selection algorithm based on ant colony optimization. Eng Appl Artif Intell 32:112–123. https://doi.org/10.1016/j.engappai.2014.03.007
31. Tabakhi S, Moradi P (2015) Relevance-redundancy feature selection based on ant colony optimization. Pattern Recognit 48:2798–2811. https://doi.org/10.1016/j.patcog.2015.03.020
32. Tabakhi S, Najafi A, Ranjbar R, Moradi P (2015) Gene selection for microarray data classification using a novel ant colony optimization. Neurocomputing 168:1024–1036. https://doi.org/10.1016/j.neucom.2015.05.022
33. Markid HY, Dadaneh BZ, Moghaddam ME (2015) Bidirectional ant colony optimization for feature selection. In: The international symposium on artificial intelligence and signal processing (AISP). IEEE, Mashhad. https://doi.org/10.1109/AISP.2015.7123519
34. Kashef S, Nezamabadi-pour H (2015) An advanced ACO algorithm for feature subset selection. Neurocomputing 147:271–279. https://doi.org/10.1016/j.neucom.2014.06.067
35. Moradi P, Rostami M (2015) Integration of graph clustering with ant colony optimization for feature selection. Knowl Based Syst 84:144–161. https://doi.org/10.1016/j.knosys.2015.04.007
36. Ghimatgar H, Kazemi K, Helfroush MS, Aarabi A (2018) An improved feature selection algorithm based on graph clustering and ant colony optimization. Knowl Based Syst 159:270–285
37. Sreeja NK, Sankar A (2015) Pattern matching based classification using Ant Colony Optimization based feature selection. Appl Soft Comput J 31:91–102. https://doi.org/10.1016/j.asoc.2015.02.036
38. (2018) Computational complexity of machine learning algorithms. https://www.thekerneltrip.com/machine/learning/computational-complexity-learning-algorithms/. Accessed 15 Feb 2019
39. Kabir MM, Shahjahan M, Murase K (2012) A new hybrid ant colony optimization algorithm for feature selection. Expert Syst Appl 39:3747–3763. https://doi.org/10.1016/j.eswa.2011.09.073
40. Fallahzadeh O, Dehghani-Bidgoli Z, Assarian M (2018) Raman spectral feature selection using ant colony optimization for breast cancer diagnosis. Lasers Med Sci 33(8):1799–1806
41. Sweetlin JD, Nehemiah HK, Kannan A (2018) Computer aided diagnosis of pulmonary hamartoma from CT scan images using ant colony optimization based feature selection. Alex Eng J 57:1557–1567
42. Yin Z, Du C, Liu J et al (2018) Research on autodisturbance-rejection control of induction motors based on an ant colony optimization algorithm. IEEE Trans Ind Electron 65:3077–3094
43. Parvin H, Moradi P, Esmaeili S (2019) TCFACO: trust-aware collaborative filtering method based on ant colony optimization. Expert Syst Appl 118:152–168
44. Uthayakumar J, Metawa N, Shankar K, Lakshmanaprabu SK (2018) Financial crisis prediction model using ant colony optimization. Int J Inf Manag. https://doi.org/10.1016/j.ijinfomgt.2018.12.001
45. Langner O, Dotsch R, Bijlstra G et al (2010) Presentation and validation of the radboud faces database. Cognit Emot 24:1377–1388. https://doi.org/10.1080/02699930903485076
46. Hamamoto Y, Uchimura S, Watanabe M et al (1998) A Gabor filter-based method for recognizing handwritten numerals. Pattern Recognit 31:395–400. https://doi.org/10.1016/S0031-3203(97)00057-5