Neurocomputing 142 (2014) 354–371

Contents lists available at ScienceDirect

Neurocomputing

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

Enriched ant colony optimization and its application in feature selection

Rana Forsati [a,*,1], Alireza Moayedikia [a,b], Richard Jensen [c], Mehrnoush Shamsfard [a], Mohammad Reza Meybodi [d]

[a] Natural Language Processing (NLP) Research Lab, Faculty of Electrical and Computer Engineering, Shahid Beheshti University, G.C, Tehran, Iran
[b] Department of Computing, Asia Pacific University, Kuala Lumpur, Malaysia
[c] Department of Computer Science, Aberystwyth University, Ceredigion, Wales, UK
[d] Department of Computer Engineering and Information Technology, Amirkabir University of Technology, Tehran, Iran

ARTICLE INFO

Article history:
Received 2 October 2013
Received in revised form
26 February 2014
Accepted 26 March 2014
Communicated by A. Belatreche
Available online 27 May 2014

Keywords:
Ant colony optimization
Feature selection
Hybrid algorithms
Swarm intelligence

ABSTRACT

This paper presents a new variant of ant colony optimization (ACO), called enRiched Ant Colony Optimization (RACO). This variation tries to consider the previously traversed edges in the earlier executions to adjust the pheromone values appropriately and prevent premature convergence. Feature selection (FS) is the task of selecting relevant features or disregarding irrelevant features from data. In order to show the efficacy of the proposed algorithm, RACO is then applied to the feature selection problem. In the RACO-based feature selection (RACOFS) algorithm, it might be assumed that the proposed algorithm considers later features with a higher priority. Hence in another variation, the algorithm is integrated with a capability local search procedure to demonstrate that this is not the case. The modified RACO algorithm is able to find globally optimal solutions but suffers from entrapment in local optima. Hence, in the third variation, the algorithm is integrated with a local search procedure to tackle this problem by searching the vicinity of the globally optimal solution. To demonstrate the effectiveness of the proposed algorithms, experiments were conducted using two measures, kappa statistics and classification accuracy, on several standard datasets. The comparisons were made with a wide variety of other swarm-based algorithms and other feature selection methods. The results indicate that the proposed algorithms have superiorities over competitors.

1. Introduction

Data preprocessing is a vital step to reduce the effect of noise and improve the quality of data processing tasks, with the aim of increasing the final efficiency of the tasks. Nowadays, real world datasets may have many irrelevant and noisy features that mislead or impede pattern recognition resulting in the discovery of finding less meaningful or even useless patterns. Through the use of feature selection, such problematic descriptors can be automatically detected and removed, resulting in more reliable pattern discovery. In addition, the availability of irrelevant dimensions in the original dataset may slow the learning process. So the reduced processing time is another benefit of FS. For example in text categorization [1], feature selection is used to reduce the size of word-document matrices and accelerates the categorization process as just the most important dimensions are considered. Feature selection also has applications in systems monitoring [2], where the most significant indices of the system are identified and only those selected indices are used to check the system performance, requiring less measurement and less computation.

In recent years many evolutionary and swarm based algorithms such as ant colony optimization [3], harmony search [4], and particle swarm optimization [5] have been utilized to tackle the feature selection problem. Ant colony optimization is a nature-inspired swarm-based approach that relies on the method that ants use to identify valuable food resources. In nature, real ants aim to find the shortest route between a food source and their nest without the use of visual information and possess no global world model, adapting to changes in the environment. One factor that the ants benefit from is pheromone deposition which enables them to reach their goal gradually. Each ant probabilistically prefers to follow a direction. The pheromone decays over time, resulting in much less pheromone on less popular paths. Given that over time the shortest route will have the higher rate of ant traversal, this path will be reinforced and the others diminished until all the ants follow the same, shortest path.

* Corresponding author. Tel.: +98 212 9904111.
[1] Part of this work was done when the corresponding author was a visiting scholar at University of Minnesota.

http://dx.doi.org/10.1016/j.neucom.2014.03.053
0925-2312/

Ant colony optimization has been considered an effective approach for finding optimal subsets in feature selection problems. The first ant colony optimization approach was presented by Dorigo, and colleagues, [6], known as Ant System (AS), in which all the pheromones are updated by all the ants which build a solution within an iteration. Another ant algorithm is Max-Min Ant System (MMAS) by Stutzle and Hoos [7] in which the pheromone values are restricted within a desired interval (e.g. $[0, 1]$) and only the global-best or iteration-best solutions are used to update the pheromone. The problems associated with these ant colony algorithms is their premature convergence after a certain number of iterations. To solve this problem, Ant Colony System (ACS), another variation, was proposed by Gambardella and Dorigo [8]. Its characteristic is that a local pheromone update is utilized to update the pheromone of the edge after an ant traversed it. The aim of local pheromone update is to diversify the exploration of the ants and make it possible for other parts of the solution space to be explored.

In this paper, a new variant of ACO is introduced, called enRiched Ant Colony Optimization (RACO). In RACO, the ants are called enriched since they consider the traversals done in the previous and current iterations. In fact the information contained in the traversals of the previous iterations is modeled as a rich source that will guide the ants' future path selections and pheromone updating stages. The purpose of considering previous traversals is to deal with the problem of premature convergence. To show the efficiency of the proposed algorithm, RACO is applied to the task of feature selection, resulting in RACO-based feature selection (RACOFS). It might be assumed that RACOFS suffers from the problem of inequality of selection in which later features have higher priorities of selection compared to earlier ones. Hence in order to show that this is not the case, RACOFS is integrated with a capability local procedure, capability RACOFS (C-RACOFS). This feature selection algorithm is a global search that is likely to become trapped in local optima. Although C-RACOFS performs a simple and superficial search in the vicinity of the globally optimal subset, it does not guarantee an appropriate improvement. Therefore it is required that the vicinity of the globally optimal solution to be searched deeply. To this end the RACOFS algorithm is integrated with an improved local procedure; this third variation is called Improver RACOFS (I-RACOFS). The main contributions of this paper is summarized below:

- A new variation of ant colony optimization (ACO) that utilizes an intelligent method for selection of edges and updating the pheromone of solutions to better guide the search process. The proposed algorithm is referred to as RACO.
- An application of the proposed RACO algorithm to the feature selection problem, as one of the most practical areas of data processing.
- An integration of the RACOFS method with a local procedure to demonstrate the capabilities of RACO in exploiting the knowledge preserved in the previous iteration traversals.
- A hybridization of RACO with an improver hybrid procedure to escape from local optima, as one of the most prevalent deficiencies of the algorithm.
- A comprehensive set of experiments on real datasets to demonstrate the merits and advantages of the proposed method and its variations in application to the feature selection problem.

1.1. Outline

The rest of the paper is organized as follows. Section 2 reviews some of the recent works on feature selection that utilizes swarm-based approaches. Also some of the applications of feature selection are reviewed. Section 3 describes the improved ant colony algorithm. In Section 4, the improved ant colony based feature selection algorithm is discussed. Section 5 presents the data sets used in our experiments, an empirical study of parameters on convergence on the behavior of proposed algorithms, and comparison of different algorithms. Finally Section 6 concludes the paper.

2. Literature review

Feature selection algorithms are mainly divided into three types: wrapper, filter and hybrid approaches. The wrapper approach involves wrapping the feature selection method with a learning model. Wrapper methods often find good subsets for a particular learning model, but incur a high computational overhead as a result of the model construction and evaluation for every considered subset. The filter approach is simpler in the sense that no model is constructed; instead, an evaluation function is used to assess the subset quality. Hence, subsets found via this approach tend to be inferior in terms of quality to wrapper algorithms while the execution of filter algorithms is faster. The hybrid approach [9,10] tries to benefit from the advantages of both methods. Hybrid methods are more time consuming than both wrapper and filter approaches, since they combine the benefits of the both algorithms. Provision of local search (i.e. helping the algorithm to escape from local optima, and tackling the entrapment problem) as a result of hybridization is one of the advantages of the hybridization. Feature selection algorithms are modeled using different sorts of optimization algorithms such as swarm intelligence (SI) [11,12,13] or evolutionary algorithms (EAs) [14] such as harmony search [15,4,16,17] or genetic algorithms [18,10]. In this section, feature selection algorithms relying on SI such as ant colony optimization, bee colony optimization (BCO) and particle swarm optimization (PSO) are reviewed and outlined in Table 1.

2.1. Swarm intelligence algorithms

Monirol Kabir and colleagues [3] proposed a new ACO based feature selection method. The algorithm considers the heuristic information of each feature as filter information while neural networks are used in the wrapper step of the algorithm. Two types of heuristic information were used for each feature, namely random and probabilistic, which have different impacts on the execution of the algorithm. The experiments showed promising results. Vieira and colleagues [19] proposed a feature selection algorithm that divides the feature selection problem into two objectives: choosing an optimal number of features and finding the most relevant features. The experiments showed good results

Table 1
Outlining the reviewed papers.

Paper	Swarm intelligence approach			Classifier
	PSO	BCO	ACO	
Kabir and colleagues [3]			√	Artificial neural network
Viera and colleagues [19]			√	Fuzzy
Jensen and colleagues [12]			√	C4.5
Ke and colleagues [20]			√	Rule-based
Chen and colleagues [21]			√	Rule-based
Forsati and colleagues [13]		√		*k*-nearest neighbor
Wang and colleagues [22]	√			Rule-based
Chuang and colleagues [23]	√			*k*-nearest neighbor
Huang and Dun [24]	√			Support vector machine
Unler and colleagues [5]	√			Support vector machine

produced from the integration of a fuzzy model classifier and the ACO algorithm. Jensen and Shen [12] proposed another algorithm that addresses the results of conventional problems associated with hill-climbing for feature selection using ant colony optimization for fuzzy-rough dimensionality reduction. Ke and colleagues [20] proposed an algorithm that integrates ACO with rough sets. The main facets of the work were the updating procedure of the pheromone trails of the edges connecting each pair of different attributes of the best-so-far solution, and also limiting the pheromone values between the upper and lower trails. As a result of solution construction and pheromone update rules, the algorithm is able to find solutions with low cardinality quickly.

Chen and colleagues [21] proposed another feature selection method that uses rough sets and ACO, which adopts mutual information as a heuristic for assessing the features' significance. The method embarks from the core (i.e. essential features) and then uses mutual information as a heuristic for feature selection. The concept of the core was first utilized in ACO in [21] such that all the ants should start with the core at the beginning of their search, and in the selection process those solutions near the core will be selected. Also other swarm-based methods exist, such as bee colony optimization. Forsati and colleagues [13] utilize the bee colony approach as one of the most recent approaches for feature selection, such that each bee produces a partial solution randomly and then returns to the hive for subsets assessment. Ultimately, the purpose is to find the most promising bees in finding solutions at the end of each iteration. The algorithm uses k-nearest neighbor classification (k-NN) along with leave one out cross validation, and outperforms some algorithms in this area.

Particle swarm optimization (PSO) [25] is an effective population-based method that has been used for many feature selection approaches in the last few years. A rough set-based binary PSO algorithm is proposed by Wang and colleagues [22] to perform attribute reduction. In the algorithm, each particle represents a potential solution, and these are evaluated using the rough set dependency degree. Chuang and colleagues [23], proposed a catfish approach that improves the binary PSO for feature selection. In this method, catfish particles start a new search when the global best value in PSO remains unchanged for three iterations. By directing PSO toward more promising regions better solutions were found. Huang and Dun [24] proposed a PSO-based feature selection method in combination with support vector machines (SVM) as the learning algorithm. Two types of PSO were combined, i.e. discrete and continuous PSO methods, for both performing the optimization of input feature subset selection and SVM kernel parameter setting concurrently. For the implementation, a distributed architecture was used using web service technology for the purpose of computational time complexity reduction. Unler and colleagues [5] proposed a new wrapper-filter method with PSO for feature selection in which PSO is used as a wrapper approach while mutual information is used as a filter approach. In fact, mutual information is used for measuring both feature redundancy and feature relevancy. Their experiments show that the algorithm is competitive in terms of computational time and classification accuracy.

2.2. Feature selection applications

Feature selection is a field of research with many applications ranging from fraud detection [26] and stock prediction [27] to advanced areas like knowledge-based authentication [28] and sentiment analysis [29]. A brief overview of some of the recent applications is given here. Tsai and Hsiao [27] proposed a system to predict the stock price through the combination of many feature selection methods to identify more representative variables for better prediction.

Duric and Song [29] proposed a new set of feature selection schemes, that rely on a content and syntax model to automatically learn a set of features. The learning process is achieved by separating the entities that are being reviewed from the subjective expressions, and describing those entities in terms of polarities. By focusing only on the subjective expressions and ignoring the entities, more salient features can be selected for document-level sentiment analysis.

Chyzhyk and colleagues [30] proposed a FS algorithm that benefits from genetic algorithms and extreme learning machines for applications in bioinformatics. The primary feature set is extracted as a voxel selection from anatomical brain magnetic resonance imaging (MRI). Voxel selection is provided by voxel based morphometry which finds statistically significant clusters of voxels that have differences across MRI volumes on a paired dataset of Alzheimer disease and healthy controls.

In another example of FS application, Chen and Liginlal [28] used a wrapper method for knowledge-based authentication. Here, the learning machine is a generative probabilistic model, with the objective of maximizing the Kullback–Leibler divergence between the true empirical distribution defined by the legitimate knowledge and the approximating distribution representing an attacking strategy that both reside in the same feature space. The experiments showed that the proposed adaptive methods performed better than the commonly used random selection method.

Ravisankar and colleagues [26] used feature selection algorithms as a tool to identify firms prone to financial statement fraud. Many techniques such as Multilayer Feed Forward Neural Network (MLFF), Support Vector Machines, Genetic Programming (GP), Group Method of Data Handling (GMDH), Logistic Regression (LR), and Probabilistic Neural Networks (PNN) are used to perform this task. The experiments were conducted using Chinese companies and revealed that PNN can outperform all the techniques without feature selection, while GP and PNN did outperform other techniques with feature selection and with marginally equal accuracies.

3. RACO: enRiched Ant Colony Optimization

The ACO algorithm [31] is a nature-inspired algorithm that simulates the natural behavior of ants, including mechanisms of cooperation and adaptation. This algorithm has been shown to be both robust and versatile, in the sense that it has been applied successfully to a range of different combinatorial optimization problems. In this section we propose a new ant colony algorithm, known as enRiched Ant Colony Optimization (RACO).

In the baseline ant colony optimization approaches, such as ant system, min-max ant system and ant colony system, the algorithms do not consider previously traversed edges and their pheromone values as a resource to guide future movements. In this paper, a new variation is proposed that considers the previous traversals as a rich source of information, to guide the explorations of the solution space to generate diverse solutions. Diversification of solutions can be achieved by increasing the exploration and exploitation abilities of the ants. To this end, during the local and final pheromone updates, the traversals of the current iterations are not only considered, but also previously traversed edges to adjust pheromones of the edges. This hypothesis is implemented through introducing the concentration rate that indicates the extent to which the algorithm should concentrate on the previously traversed edges or the traversals of the current iteration.

Fig. 1 shows the stages of the algorithm that benefit from the information from traversals. Here, the algorithm uses two datasets of current and previous traversals. Different stages of the algorithm benefit from these stored data. In the selection stage, the

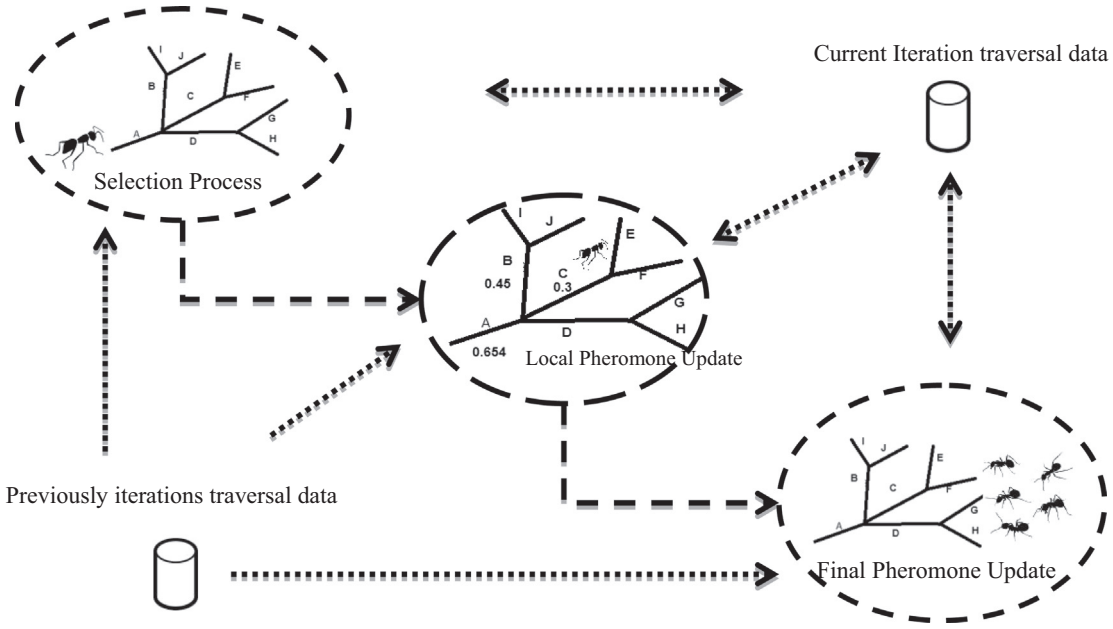

Fig. 1. The general process of the algorithm (doubled head dotted arrows shows the two way exchange of data, while one-direction dotted arrows show one way exchange of data, and finally dashed arrows show the flow of the algorithm).

ants consider the previous tours taken by ants to select an appropriate edge. In the local pheromone update phase, the databases are used to lay an appropriate pheromone value on the edge and finally in the final pheromone stage the traversals information is used for the global update of the edges. The interactions with the previous traversal database is unidirectional in the sense that the ants are only allowed to use the previous traversals' information without manipulating them, while the current database traversals can be either read from or manipulated by the ants, in the sense that the pheromone in the current database can be updated. After finishing the current iteration, the data that resides in the current traversal database will be added to the previous traversals' database.

ACO algorithms generally contain the following steps.

- Step 1: Initialization.
- Step 2: Solution creation.
- Step 3: Solution evaluation.
- Step 4: Pheromone update.

RACO starts with the initialization step. In this step, the algorithm's dynamic parameters are initialized. These parameters include the number of ants (A_T), the number of generations (I_T) and the initial pheromone values of each edge (γ). Also, each ant randomly selects its initial state.

After successfully initializing the RACO algorithm, the process starts with creation of solutions. Each ant creates its own partial solution independently. During the solution creation phase, each ant selects an edge and then updates the pheromone of the same traversed edge. The ants choose a path according to the selection probability which helps them to identify worthwhile paths. The calculation of the selection probability is shown in Eq. (1).

$$Selection\ Probability_{ab} = \begin{cases} \varphi_c^{ab} \times Popularity\ Rate_{ab} & Iteration > 1 \\ \varphi_c^{ab} & Iteration = 1 \end{cases} \tag{1}$$

$$Popularity\ Rate_{ab} = \frac{\sum_{j=1}^{N} traversals_j(a,b)}{\sum} traversals \tag{2}$$

where φ_c^{ab} is the pheromone value laid on the edge (a, b) and $PopularityRate_{ab}$ is the rate of popularity of the edge (a, b) among the ants. Here, A_T is the total number of ants, N is the total number of ants that have traversed edge (a, b), $\sum_{j=1}^{N} traversals_j(a,b)$ is the total number of traversals that include edge (a, b), and finally $\sum traversals$ is the total number of traversals by the ants. We try to measure how many ants have traversed the edge (a, b) in proportion to the total traversals, within the current and previous traversals. The more the edge (a, b) is selected its selection probability increases. Merely considering popularity rate will not help to prevent premature convergence, and on the other hand will encourage ants to select edges that are selected frequently. Therefore, the current pheromone φ_c^{ab} of the edge (a, b) should be considered to reflect the importance of the edge in comparison to its neighbors and generally in the solution space, to measure how useful an edge is. For example, in the calculation of the probability of selection of the edge (b, c), if in total the ants have traversed 183 edges so far and the edge (b, c) has appeared 27 times in their traversals, then the popularity rate will be 0.147. This value multiplied by the previously laid pheromone value of the edge will give the selection probability. Eq. (2) requires the availability of previous traversals information, while in the first iteration ($Iteration = 1$ as shown in Eq. (1)) this information is not available. Therefore in the first iteration, the ants choose the edges according to pheromone values only.

Eq. (1) would make the selection of the edges strongly dependent on the initial pheromone values, in the sense that an edge with initially high pheromone will have high probability of selection in future traversals, and consequently will lead to premature convergence. Hence a local pheromone update is required to increase the exploration ability of the ants and prevent premature convergence. Eq. (3) is the local pheromone update policy:

$$\varphi_n^{ab} = \begin{cases} \varphi_c^{ab} & \Delta\varphi = 0 \\ \Delta\varphi & \Delta\varphi - \varphi_c^{ab} > 0 \\ \varphi_c^{ab} - \Delta\varphi & \Delta\varphi - \varphi_c^{ab} < 0 \end{cases} \tag{3}$$

where φ_c^{ab} and φ_n^{ab} is the current pheromone value and the new pheromone value of the edge (a, b) and $\Delta\varphi$ is defined according to

Eq. (4). The purpose is to decrease the probability of selection of the frequently traversed edges, and consequently increase the exploration ability. Hence the pheromone value of an edge with high rate of traversals should be reduced.

$$\Delta\varphi = \begin{cases} \alpha \log\left(\frac{N(s)}{N_{ab}(s)}\right) + (1-\alpha)\log\left(\frac{TN(s)}{TN_{ab}(s)}\right) & Iteration > 1 \\ \log\left(\frac{N(s)}{N_{ab}(s)}\right) & Iteration = 1 \end{cases} \quad (4)$$

here, $N_{ab}(s)$ and $N(s)$ (or $TN_{ab}(s)$ and $TN(s)$) are the number of traversals involving (a, b) and total traversals within the current iteration (or the number of traversals involving (a, b) and total traversals within the previous iterations), respectively. To analyze Eq. (4), by increasing the difference between $N_{ab}(s)$ and $N(s)$ (or $TN_{ab}(s)$ and $TN(s)$) the logarithm parameter becomes larger, which indicates that the edge (a, b) has been visited less than other edges. Therefore, the edge is prone to be visited more, and more pheromone should be laid on it. In contrast, if the difference becomes lower, the pheromone value should be decreased. This fact will increase the ability of the ants to explore the solution space. However, an exceptional case occurs when $N(s)=N_{ab}(s)$; for instance, when the first ant in the first iteration traverses the first edge, while by increasing the number of traversals, $N(s)$ and $N_{ab}(s)$ will no longer be equal. In this case $\Delta\varphi = 0$ since the logarithm's input is one. Also in the first iteration the traversals of the current iteration and the previous traversals are the same (i.e. $N(s) = TN(s)$ and $N_{ab}(s) = TN_{ab}(s)$).

The parameter $\alpha \in [0\ldots1]$ is the concentration rate which plays the role of the pheromone decay coefficient, and is a variable which identifies the extent to which the algorithm should focus on within-iteration traversals or total traversals of the previous iterations. The higher this value is, the more emphasis there will be on the current iteration's traversals. As all the ants created their solutions, the quality of each solution should be checked. This assessment can be done using a given fitness function that satisfies the algorithm's objective.

The last step of the ant colony algorithm is the final pheromone update in which all the ants are allowed to update only the edges that they have traversed, but the effects of their updates on the same edge are not identical. In simpler terms, those ants with a higher value of fitness f_x, can increase the pheromone of the edge (a, b) (if this edge is included in their traversal) more than those ants that have traversed this edge but their fitness is lower than f_x. To implement this idea we propose a final pheromone equation which involves three main parts:

$$Finale\ Pheromone\ Update_{ab} = Contribution\ Rate_{ab} \times Popularity\ Rate_{ab} \times Average\ Weights_{ab} \quad (5)$$

As shown in Eq. (5) the final pheromone of an edge indicates its relative importance in the solution space. Hence, three factors can influence the pheromone value of an edge. The more an edge is popular, the more useful it is. Therefore the popularity rate of an edge should be measured as explained in Eq. (2). In addition, the frequency of selection does not necessarily mean high importance, since frequent traversals of an edge might lead to poor results. Therefore the level of contribution of a given edge (a, b) towards the fitness function should be taken into account to accurately adjust the pheromone values. The *ContributionRate* is the average of the fitness values of those ants that have the edge (a, b) in their traversal, normalized by dividing by the total fitness. The aim is to measure the contribution of the edge (a, b) by looking at the fitness values of the ants.

$$Contribution\ Rate_{ab} = \frac{\sum_{j=1}^{N} F_j}{\sum_{i=1}^{A_T} F_i} \quad (6)$$

here, A_T is the total number of ants and N is the total number of ants that have traversed edge (a, b). F_j is the fitness value of the ant that has edge (a, b) in its traversal, and F_i is the fitness value of the ith ant. If the frequency of selection of an edge leads to poor results, then the ants having this edge in their traversals will mostly have low fitness values on average, and finally will lead to a lower contribution rate.

To reflect the importance of an edge in comparison to other edges in the solution space, the average of the pheromone values assigned to an edge is considered. In fact, if an edge is significant in comparison to its neighborhood edges, its average is higher than the others, and consequently the pheromone value increases. In the local pheromone value update, the aim is to reduce the pheromone of the frequently visited edges, while increasing the pheromone value of the rarely visited edges, to increase the exploration ability. In the final pheromone update the aim is to increase the pheromone of the worthwhile edges to increase the exploitation ability of the ants.

$$Average\ Weights_{ab} = \frac{\sum_{u=1}^{N} pheromone_u^g(a, b)}{||N||} \quad (7)$$

Eq. (7) calculates the average of the changes made in the pheromone values of the edge (a, b) during the gth iteration. The higher the value of *AverageWeights*, the more pheromone will be laid on the edge (a, b). In Eq. (7), $pheromone_u^g(a, b)$ is the local pheromone value that was laid on edge (a, b) in the gth generation when the uth ant traversed it, and N is the number of ants that have edge (a, b) in their traversals.

4. Feature selection with RACO

In this section we propose a new RACO-based feature selection method, RACOFS. In this algorithm, solutions are encoded as a string of serial bits of 0*s* and 1*s*, in which 1 indicates a selected feature and 0 an ignored feature. For instance, Fig. 2 shows that the first, sixth and the seventh features are selected, and the other features are unselected.

The first step is the initialization of the algorithm. In the initialization stage the number of ants and iterations are chosen by the user. Then each ant is randomly assigned to a number between 1 and F as its initially selected features, in which F is the total number of features. Additionally, the pheromone on each edge is initialized randomly.

Algorithm 1. Selection probability rule.

Input:
Initially selected feature of an ant
Output:
Created solution of an ant
Algorithm:
while *true*
Generate a probability number P_n using selection probability equation
Select the edge with smallest pheromone that is bigger than, P_n
if the selected edge leads to the last feature
Select the last feature;
break;
else
Select the feature;

1	0	0	0	0	1	1

Fig. 2. Solution representation.

```
end if
if the selected edge leads to feature F-1
   Generate a probability number P_l using selection
probability equation
   if P_l is bigger than the pheromone of the edge leading to
feature F
      Select the last feature.
      break;
      end if
   end if
end while
```

Using the selection probability relation (Eq. (1)), the ants will select their next feature to select. In order to prevent from some common mistakes such as a non-stop loops of traversal and edge selection, in this algorithm if an ant chooses the ith feature, it then cannot choose the jth feature if $j<i$. If the selected feature's number does not make it possible for the ants to proceed further (i.e. one before the last feature is selected), then no further movement is allowed; that is the point the solution construction for the ith ant is completed and the solution should be evaluated.

The problem occurring in this type of selection is that in most cases, the last feature can be selected if the feature $F-1$ is also selected. Therefore if the number of the currently selected features is equal to $F-1$ then the last feature is selected only if the selection probability value is bigger than the pheromone connecting the two last features to each other. Algorithm 1 shows the selection probability rule of the algorithm. After traversing an edge, the local pheromone value is updated according to the local pheromone update policy (Eq. (3)). Finally after the generation of each solution by each ant, the solutions must be assessed to identify their goodness. The last stage is the local pheromone update in which the edges connecting features are updated based on the final pheromone rule discussed in Eq. (5). Since the outcome of selection probability of each edge is a number within the interval [0,1] it is likely that the deposited pheromone will become greater than 1 during the pheromone update stages. To overcome this problem the pheromone value of each edge is normalized; the pheromone values on each edge that connects a parent to its children should sum to one. The algorithm iterates for I_T number of iterations. The complete process of RACOFS is shown in Algorithm 2.

Algorithm 2. RACOFS.

```
Input:
      Number of iterations and ants
      α value
Output:
      The best ant in terms of fitness
Algorithm:
  while G number of iterations are not finished
    Initialize the ants' first movement.
    foreach ant in the i-th iteration
      Select the next feature according to selection probability
      rule
      Update the pheromone laid on the last traversed edge
      using local pheromone update rule
      Apply the pheromone normalization step
        if further movement for the i-th ant is impossible
          Assess the fitness for the generated solution.
    end for
  Apply final pheromone update
  Apply the pheromone normalization
end while
```

4.1. Hybrid algorithms

It might be assumed that RACOFS suffers from an inequality of selection in the sense that the first features have a lower probability of selection while the later features will have higher priorities. Hence, a local procedure is integrated with RACOFS to:

- Evaluate this assumption of inequality of selection.
- Show that the reliance on previous traversals is effective to distinguish relevant and irrelevant features to improve the quality of the solutions.

Since the primary aim of using this hybrid procedure is to show the capability of RACOFS this variation is called capability RACOFS (C-RACOFS). According to Algorithm 3, in C-RACOFS after the generation of a solution all the features are tested to see if there is an improvement in subset quality by their removal or addition. A selected feature will be changed temporarily to an unselected one, while an unselected feature is changed temporarily to a selected one. If the fitness of this solution is better than the fitness of the older solution, then the new solution will replace the older one. The aim is to determine if RACOFS has included or excluded a feature mistakenly in the final subset.

Algorithm 3. C-RACOFS

```
Input:
     A set of solutions created by the ants
Output:
     Improved set of solutions
Algorithm:
foreach solution created by each ant
  foreach feature
    if the feature is not selected
      Change it to a selected one;
      Assess the fitness of this new solution S_n;
      if the S_n is better than the older solution S_d
          Replace S_n with S_d;
  else
    Change it to a selected one;
    Assess the fitness of this new solution S_n;
    if the S_n is better than the older solution S_d
        Replace S_n with S_d;
      end if
    end for
  end for
```

The capability hybrid algorithm should not be considered as a greedy search algorithm. For greedy search algorithms, in a solution space with the size of n, there will be 2^n number of different combination of the features, and correspondingly 2^n number of different solutions, while in this hybrid algorithm for a solution with n number of features, $n+1$ number of different solutions is available (including original solution). As shown in Fig. 3, each selected/unselected bit is changed to an unselected/selected one.

Ant colony optimization is effective in performing global search and finding the approximate region of the globally optimal solution, but suffers from entrapment in local optima. Therefore its hybridization with a local search procedure is inevitable, to improve the final results. According to Fig. 3, if the original solution is considered as globally optimal, then changing the bits iteratively to check further improvements would be a kind of superficial local search in the sense that a small vicinity of the

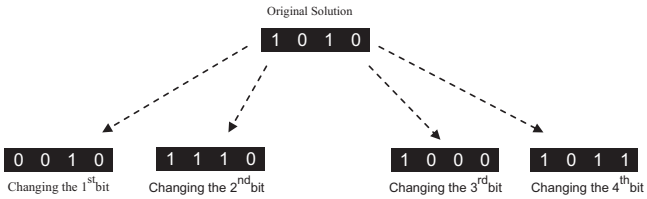

Fig. 3. C-RACOFS graphical illustration.

globally optimal solution is searched for possible improvements. However this type of local search is superficial and does not guarantee to be applicable enough. Therefore RACOFS should be integrated with another local search procedure which not only searches deeper vicinities of the globally optimal solution, but also ensures superiority over RACOFS and C-RACOFS.

The algorithm ImprovementProcedure, as shown in Algorithm 4, is another local search algorithm that is applied to RACOFS that ensures better search around a good solution. This local search procedure is also applied to GA-based feature selection [10] and showed good performance in improving solutions produced by simple genetic algorithms. Hence this local search is applied to RACOFS and named as Improver RACOFS (I-RACOFS) since it aims at improving RACOFS.

I-RACOFS heavily depends on the atomic operations of ripple_rem(r) and ripple_add(r) and a prespecified subset size (d), for its execution. The ripple_rem(r) operation removes r number of least significant features and adds $r-1$ of the most significant features. On the other hand ripple_add(r) adds r of the most significant features while removing the $r-1$ least significant features. The procedure of adding and removing iterates until the condition $|X|=d$ is met. In Algorithm 4, three scenarios might occur:

- *Scenario 1*($|X|=d$): Then ripple_add(r) and ripple_rem(r), will add r of the most significant features and remove r of the least significant features, respectively.
- *Scenario 2*($|X|>d$): ripple_rem removes r of the least significant features, while ripple_add removes $r-1$ of the most significant features.
- *Scenario 3*($|X|<d$): ripple_add adds r of the most significant features, while ripple_rem removes $r-1$ of the least significant features.

A feature is least significant if the level of its contribution toward the quality of solution in comparison to other features is low (i.e. by removing the feature from the original subset the quality of the solution does not decrease much). A feature is most significant if the level of its contribution toward the quality of solution in comparison to other features is high (i.e. by removing the feature from the original subset the quality of the solution decreases).

Algorithm 4. Improvement Procedure.

Input:
A solution, G_{sol}
Ripple factor: r.
Desired subset size, d.
Output:
A locally improved solution.
Algorithm:
Put selected features of solution S in the set X
Put unselected features of solution S in the set Y
if $|X|=d$
Select r of the most significant feature from the set Y
Remove r of the least significant features from the set X
if $|X|>d$
while $|X|$ and d are not equal
Select $r-1$ of the most significant feature from the set Y
Remove r of the least significant features from the set X
end while
end if
if $|X|<d$
while $|X|$ and d are not equal
Select r of the most significant feature from the set Y
Remove $r-1$ of the least significant features from the set X
end while
end if

I-RACOFS is detailed in Algorithm 5. First, the ordinary RACOFS is performed and then the best solution is passed to the local search procedure. r and d are the ripple factor and the desired number of features, respectively.

Algorithm 5. I-RACOFS

Input:
A set of solutions created by the ants
Output:
A set of solutions improved by the I-RACOFS
Algorithm:
while iterations are not finished
BestSolution = RACOFS()
ImprovedSolution = ImprovementProcedure (BestSolution, r, d)
Replace ImprovedSolution with the BestSolution
end while

In cases that the desired subset size (d) is equal to the size of the currently generated solution (i.e. $|X|=d$), then r of the least significant features are removed and r of the most important features will be added, while in the two other cases the difference between addition and subtraction is always one, and the algorithm iterates as long as the subset size becomes equal to d. For further details regarding the performance of ripple factors (i.e. ripple_rem and ripple_add) interested readers can refer to [10]. Table 2 defines the parameters of the algorithm.

4.2. Timing analysis and memory consumption

In this section we investigate the time complexity and memory consumption rates of the proposed methods. The required parameters and the symbols of the proposed algorithms are shown in Table 2. For RACOFS, each ant only evaluates one subset; therefore in the worst case where all ants select all features each ant will perform F evaluations. If there are A_T ants, then timing complexity is $O(A_T \times F)$. This is repeated over I_T generations, giving a complexity of $O(I_T \times A_T \times F)$. For the C-RACOFS method, the fitness is calculated only for 0s. Its worst-case is $O(I_T \times A_T \times F + I_T \times F \times F)$, for each feature 0 appearing in the solution (worst case F), then the subset is re-evaluated (taking F time).

The general timing analysis for I-RACOFS operator is difficult, as it is unknown how many 0s will appear in any given subset. The worst case is where all bits are 0. We know that for a subset of size s, $(F-s)$ bits will be zero, so the fitness is evaluated $(F-s)$ times. Also, for I-RACOFS, the local search only considers the

Table 2
Parameter definition of RACOFS and its variations.

Parameter	Definitions	Variations		
		RACOFS	Hybrid	
			C-RACOFS	I-RACOFS
A_T	Number of ants	√	√	
I_T	Number of generations	√		
γ	Pheromone table	√		
r	Ripple factor			√
d	Desired subset size			√
F or D	Total number of features in the dataset	√	√	√
N	Number of ants have traversed a specific edge (e.g. (a, b))	√		
φ_c^{ab}	Current pheromone value of the edge (a, b)	√		
φ_n^{ab}	New pheromone value of the edge (a, b)	√		
α	Concentration rate	√		
X	Set of selected features			√
Y	Set of unselected features			√

addition of single features, by considering feature elimination the worst-case complexity would be the same, though it would take a bit more time on average. But I-RACOFS differs from the others. In Algorithm 3 the subset size should be known prior to timing analysis. Some papers [10] have used big-O or number of subset evaluations, but they are not helpful because subsets with different sizes may produce different amounts of time computation.

Since the proposed algorithms rely on the previous traversals in the previous iterations, some discussions regarding the memory consumption rates might be necessary. Considering a solution space with the size of F, there will be $(F(F-1)/2)$ number of edges connecting each node to all other nodes in the space. If preservation of each pheromone value consumes M bytes of memory, then the total memory consumption that preserves the information of a given iteration will be $M \times (F(F-1)/2)$ bytes. Consequently for I_T number of generations the memory consumption rate will be $M \times I_T \times (F(F-1)/2)$. I_T and M are constant values hence the memory consumption will be highly dependent on F, the number of features in a dataset.

5. Experimental results and discussions

In this section the proposed algorithms are evaluated and compared with a wide range of the other state-of-the-art algorithms. We compare our work with a wide range of other related works, such as swarm intelligence algorithms including ant colony optimization algorithms implemented as feature selection such as [6,7,8,32], and other swarm-based feature selection algorithms including bee colony [13], PSO [5] and ant colony [3]. In [3] the authors have proposed two variations, random and probabilistic, and these two variations in this paper are named as ACOFS-R, and ACOFS-P, respectively. Also comparisons with other non-swarm algorithms are made, such as genetic [10] and the baseline FS algorithms.

Two well-known measures of classification accuracy (CA) and kappa statistics (KS) are used to show the inferiorities and superiorities of the proposed works. CA is introduced in Eq. (8), where TotalSamples is the number of instances in the dataset and, correctly classified samples are the number of samples whose class was predicted correctly.

$$Classification\ Accuracy = \frac{Correctly\ Classified\ Samples}{Total\ Samples} \times 100 \qquad (8)$$

The other measure used is the kappa statistic [33], which is a prevalent statistical measure that models the performance and allows for input sampling bias. This measure has been used in many feature selection methods e.g. [34,5,35], etc. The aim of using this measure is to assess the level of agreement between the classifier's output and the actual classes of the dataset. The kappa statistic can be calculated as follows:

$$P(A) = \frac{\sum_{i=1}^{c} N_i}{N} \qquad (9)$$

$$P(E) = \sum_{i=1}^{c} \left(\frac{N_{i*}}{N} \times \frac{N_{*i}}{N} \right) \qquad (10)$$

$$kappa = \frac{P(A) - P(E)}{1 - P(E)} \qquad (11)$$

where N is the total number of samples and N_i is the number of samples in a data set which are correctly classified by the classifier. In Eq. (10), N_{i*} is the number of instances recognized as class i, by the classifier and N_{*i} is the number of instances belonging to class i in the dataset. The purpose is to maximize this measure. Finally kappa can be measure as Eq. (11) in which kappa $\in [0,1]$, kappa$=0$ and kappa$=1$, means there is not agreement between classifier and the actual classes, and perfect agreement on every example, respectively. In the rest of this section the datasets are introduced then some experiments are done which justifies the selection of the classifier. The hybridization parameters of ripple factor (r) and the desired subset size (d) are tested to investigate the effects of these parameters on searching the solution space and generally on the behavior of the proposed variations. Comparisons and the timing analysis of the proposed algorithms constitute the last two subsections.

5.1. Datasets

The used datasets are shown in Table 3. All of the datasets were downloaded from the UCI Machine Learning Repository[2]. Based on the categorization by UCI, datasets are divided into three categories of small (dimension equal to or smaller than 10), medium (dimension between 10 and 100) and large (dimension equal to or greater than 100). The first three columns are related to the datasets description, while the last three columns outline the size and dimensions of the datasets. The middle column, concentration rate, indicates the extent to which the proposed algorithms relied

[2] http://archive.ics.uci.edu/ml/datasets.html.

Table 3
UCI datasets.

Category type	Dataset	Symbols	Concentration rate (α)	No. samples	No. features	No. classes
No. features ≤ 10 small)	Monk1	MK1	0.15	124	6	2
	Monk2	MK2	0.05	169	6	2
	Post-operative	PO	0.05	90	8	2
	BreastCancer	BC	0.2	699	9	2
	Glass	GL	0.5	214	10	7
	Vowel	VW	0.1	990	10	11
10 $<$ no. features $<$ 100 (medium)	Wine	WI	0.95	178	13	2
	Zoo	ZO	0.25	101	17	10
	Horse	HR	0.85	368	27	2
	Ionosphere	IO	0.25	351	34	2
	Soybean-small	SS	0.05	35	47	4
	Sonar	SO	0.55	208	60	2
No. features ≥ 100 (large)	Arrhythmia	ARR	0.02	452	279	16
	Hill-Valley	HV	0.02	606	101	2

Table SEQ Table * ARABIC 4
Parameter setting of the implemented algorithms[a].

Algorithms		Parameter settings
Proposed algorithms	RACOFS	$I_T=10$, $A_T=100$, $\gamma=$randomly between 0 and 1
	C-RACOFS	$I_T=10$, $A_T=100$, $d=$based on the dataset size, $r\in$ [1–4]
	I-RACOFS	Iterations=10, ants=100, pheromone decay coefficient=0.1, evaporation_Rate=0.3, $q=3$, $\dot{\rho}=0.7$, $\alpha=1$, $\beta=3$
Baseline ACO algorithms	AS	Iterations=10, ants=100, pheromone decay coefficient=0.1, evaporation_Rate=0.3, min_pheromone=0.5, max_pheromone=1, $q=1$, $\dot{\rho}=0.9$, $\alpha=1$, $\beta=3$
	MMAS	Iterations=10, ants=100, pheromone decay coefficient=0.1, evaporation_Rate=0.3, $q=1$, $\dot{\rho}=0.9$, $\alpha=1$, $\beta=3$
	ACS	$\alpha=1$, $\beta=6$, $r=5$, $\tau_{max}=0.999$, $\tau_{min}=0.001$, $m_1=0.15$, $m_2=1$, $\dot{\rho}_1=0.01$, $\dot{\rho}_2=0.03$
Recent ACO algorithms	TSIACO1	$\alpha=1$, $\beta=6$, $r=5$, $\tau_{max}=0.999$, $\tau_{min}=0.001$, $m_1=0.15$, $m_2=0.5$, $c_1=0.9$, $\dot{\rho}_1=0.002$, $\dot{\rho}_2=0.03$
	TSIACO2	$\alpha=1$, $\beta=3$, $\rho=0.4$, $\tau=0.5$, $\mu=0.08-0.6$, $\eta=0.1$, $\phi=0.1$, $ants=100$
	ACOFS-R	
	ACOFS-P	
BCO	BCOFS	Bees=100, iterations=10, NC=depends on the dataset size

[a] For the implemented algorithms the parameter settings were done based on the settings proposed in the reference papers (except number of ants and iterations), while baseline algorithms were fine-tuned to the most optimal results.

on the traversals of the current iteration. This parameter is fine-tuned for each dataset separately.

5.2. Classifier performances

In this section some experiments are carried out using RACOFS variations to show the effectiveness of *k*-NN for our algorithms. One of the reasons behind the utilization of *k*-NN is the type of datasets that have been used. As we have conducted our experiments on datasets having more than two class labels, *k*-NN in such cases would be reasonable, easier and of higher utility compared to other classifiers such as support vector machine, as SVM classifiers utilization for samples having more than two class labels is intractably difficult. The use of other types of classifier (e.g. CART) does not seem reasonable, as these rely on a training process to construct the classifier, to then be able to classify test samples. In this example, some datasets that are intrinsically divided into two disjoint groups of train-test (e.g. MK1, MK2 and HV) might benefit from CART. The naïve Bayesian classifier (or simply Bayesian) can be seen as another suitable classifier. Hence its performance is tested against *k*-NN in three datasets of BC, HR and HV; each representing a category of small, medium and large datasets, respectively. The performance comparisons are made in terms of accuracy and timing execution, under similar conditions as outlined in Table 4.

Fig. 4 compares the *k*-NN and Bayesian classifiers in terms of classification accuracy. The testing conditions, in terms of the number of ants and iterations are the same for both classifiers, while in the Bayesian classifier we used the equiprobable

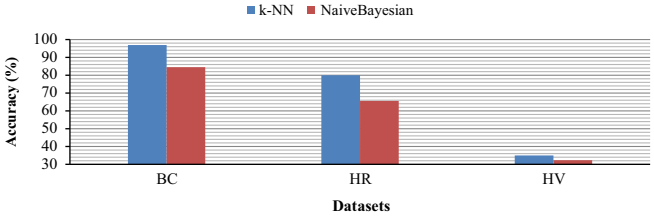

Fig. 4. *k*-NN and naïve Bayesian comparisons on three representative datasets in terms of accuracy.

partitioning technique [3,36] for data partitioning. According to the experiments illustrated in Fig. 4, *k*-NN classifies the samples better than the Bayesian classifier in all three categories. The inferiority of the Bayesian approach could be as a result of the data partitioning stage, as partitioning makes the data more general, resulting in a loss of useful information, while *k*-NN considers the actual, unchanged, data during classification.

The other comparison criterion is the execution time. In Fig. 5, the aim is to show the execution time of *k*-NN and naïve Bayesian classifiers for different subset sizes, using a 2-class dataset. Theoretically, by increasing the number of instances of a dataset, the *k*-NN execution time increases. Also, *k*-NN relies on the distances between samples (e.g. Euclidean distance) for classification. Hence increasing the number of available dimensions (features) of the dataset prolongs the distance measurement calculation time. Therefore, the *k*-NN execution time depends on the subset size and the number of instances. Bayesian classifiers also can be affected by the number of instances of the dataset, but

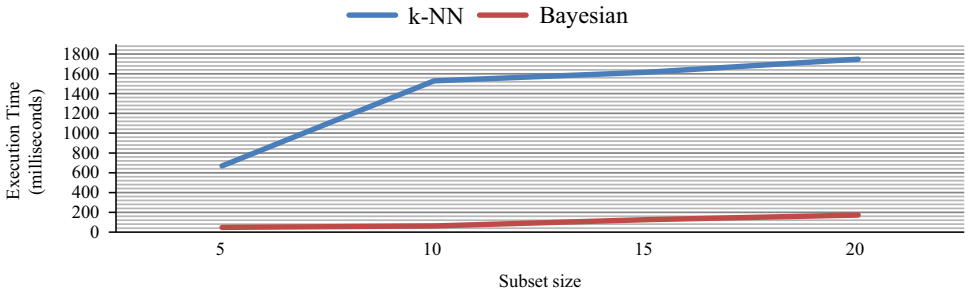

Fig. 5. Timing analysis of *k*-NN and Bayesian classifiers in a 2-class dataset for different subset sizes.

do not use any distance measure in the classification process. Hence the use of a distance function prolongs the execution of *k*-NN compared to naïve Bayesian, but assures better results.

Hence *k*-NN is used as the main classifier. However fine-tuning of the parameter *k*, is necessary. In Fig. 6, some experiments are carried out to study the effects of this parameter. As expected, by increasing the number of neighbors the accuracy of classification increases. This behavior stems from the fact that increasing the value of *k* would prevent over-fitting to a certain extent.

In this paper the value of *k* has been set to 1 in order to have a fair and consistent comparison with the literature [13,10]. In datasets that are intrinsically divided into two sets of training and testing such as MK1, MK2 and HV the experiments were carried out in this form with 1-NN as classifier, instead of LOOCV and *k*-NN, in which $k=1$.

5.3. Effects of hybridization parameters

In Figs. 7–9, the effects of the ripple factor (r) and the desired subset size (d) are tested. The proposed hybrid algorithm of I-RACOFS relies heavily on these two parameters. Therefore in this section, some experimentation is given to show the effects of increasing or decreasing the ripple factor and the desired subset size on the algorithms behavior, using three deliberate datasets of SO, HR and HV. The appropriate values of r in all the datasets are 1, 2, 3 and 4, and for $d=$D/5, 2D/5, 3D/5 and 4D/5, as used in [10]. Ripple factors cannot be applied when $r > d$ [10].

- Ripple factor affects the fitness value of a solution.
  This point indicates that the ripple factor makes the search in the local optima stronger that would lead to improved fitness value. However, this may not be the case for all datasets and depends on the dataset characteristics like number of dimensions or instances. For instance in Fig. 8, when $d=24$ the fitness for $r=3$ is around 94% while the same rate of accuracy was preserved for higher amounts of ripple factor (e.g. $r=2$, 3 and 4).
- Always increasing/decreasing the subset size and ripple factor is not effective.
  For example, in Fig. 7, the best result is obtained when $d=12$ and $r=4$. But when the value of d is too small (e.g. $d=6$) or too large (e.g. $d=24$), the ignored or added features have negative effects on the classification accuracy, in the sense that the newly added or ignored features will decrease the accuracy. The solution space size can affect the selection of ripple factor as well. The selection of the value of the ripple factor is critical in the sense that choosing low rates for this would decrease the searching ability of the hybrid procedure while very high values of the ripple factor results in a time-consuming algorithm. Therefore, it is likely that in solution spaces with low dimensions, small values of r lead to an algorithm with satisfactory results, while by increasing the size of the solution space, it is less likely to reach an optimal solution when r is small. Similarly for datasets with high numbers of features like datasets with size greater than 100 (Fig. 9), it is needed to define larger subset sizes (e.g. 3D/5 or 4D/5), while in small or medium-sized datasets, the selection of a low subset size (e.g. D/5 or 2D/5) would help to attain to an optimal solution.

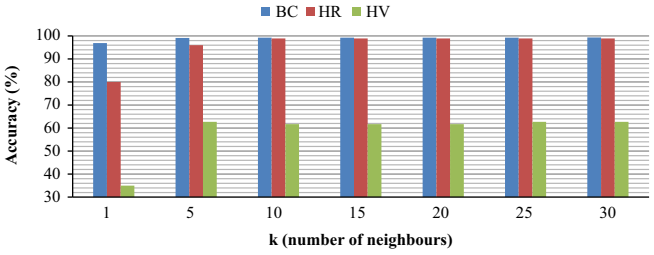

Fig. 6. Studying the effects of parameter *k* in *k*-NN classifier on three representative datasets.

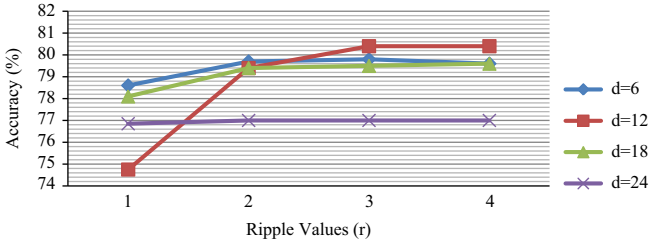

Fig. 7. The effect of ripple factor for the HR dataset.

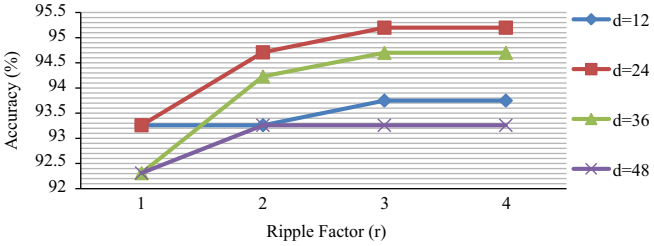

Fig. 8. The effect of ripple factor for the SO dataset.

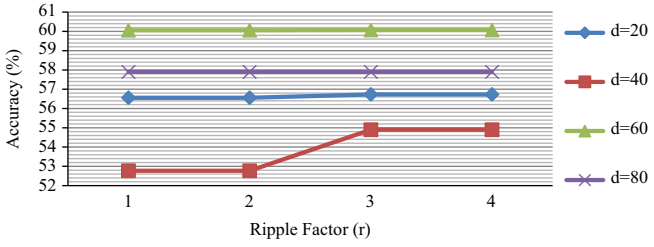

Fig. 9. The effects of ripple factor for the HV dataset.

- Effects on convergence criterion.
  The higher the ripple value is, the sooner the algorithm converges. The convergence of an ordinary global search algorithm would take place after many iterations, while in this hybrid algorithm the convergence to the optimal result would be faster with a suitable ripple factor.

5.4. Comparisons

In this section of the experimental results, the performance of the proposed algorithms is compared to some of the most recent swarm intelligence algorithms, proposed either for FS or non-FS tasks. The comparisons are divided into two sections. In the first subsection, swarm-based algorithms are compared, and the second section contains some comparisons with genetic and baseline feature selection algorithms. Parameters of the implemented algorithms are set according to Table 4.

In all algorithms the number of ants and iterations are set to the ones used in this algorithm (RACOFS). The algorithms were tested in ten independent executions for justifiable comparisons. In BCOFS [13], the number of constructive steps (NC) has different values based on the dimensions of the dataset. NC for small datasets is D/3, and for medium and large datasets NC is equal to D/5 and D/10, respectively.

5.4.1. Swarm-based comparisons

In this section we compare the proposed RACOFS algorithms with the ant and bee colony algorithms proposed in [32] and [13], respectively. The comparisons reveal significant superiorities over competitors. In Tables 5–7, the results are shown in the form of x–$y(z)$, where x, y and z are the average of CA, KS values and the subset size of the best solution, respectively.

In Table 5, the comparisons are made for small datasets. Ant colony comparisons are divided into two types of baseline [6–8] and recently proposed variations [32,3] in which the algorithms are implemented as feature selection algorithms while the other ant-based feature selection method [3] was implemented and tested in more datasets, using LOOCV and 1-NN. BCOFS [13] is our previously proposed algorithm.

For the MK1 and MK2 datasets I-RACOFS compared to two other variations of RACOFS and C-RACOFS could not show satisfactory results, as it ignored an optimal subset size gained by RACOFS and C-RACOFS. In the MK 1 and MK2 datasets, the optimal subset sizes are three and six, respectively. For these datasets RACOFS is also superior to other algorithms of ant colony and bee colony as compared in Table 5.

For the PO dataset, the performances of I-RACOFS and RACOFS are the same for $d=2$, while by increasing the desired subset size in I-RACOFS the performance deteriorates. Also RACOFS could outperform other variations of ant colony but was inferior to BCOFS. For the BC dataset, the best result in terms of CA was gained by RACOFS with the size of four, while I-RACOFS could outperform other variations when $d=4$, in terms of the KS measure. For the GL dataset the proposed variations did outperform the competitors. For the VW dataset, in the proposed variation the best result was gained by RACOFS with the size of eight. However this algorithm did not outperform baseline ant algorithms. In I-RACOFS other variations, except baseline ACOs, are outperformed in terms of the KS measure only.

Determination of the desired subset size would be one of the most important factors of inferiorities of I-RACOFS over the other variation, while in some datasets for specified values of d and r, I-RACOFS outperformed the competitors. Also, the partial reliance of RACOFS on previous traversals would be the most prominent factor leading to the algorithm's superiority over its competitors.

In Table 6, the proposed variations are compared with other swarm-based algorithms for medium-sized datasets. In RACOFS experiments in the WI dataset, RACOFS and its variations could not show any superiority over other algorithms and it is likely that the variations in this dataset suffer from the classifier settings of the proposed algorithm, as RACOFS uses k-NN with $k=1$, while this dataset may require $k>1$. In the ZO dataset, RACOFS is superior over the competitors in terms of both CA and KS measures.

In the HR dataset, I-RACOFS outperformed all other algorithms, including RACOFS and C-RACOFS, when $d=12$ and $r>2$. In the SS dataset, the results of the algorithms are the same while the differences are in the subset sizes, in which RACOFS is better. In two other datasets, IO and SO, the hybridization is more significant. I-RACOFS could improve both variations of RACOFS and C-RACOFS in the SO and IO datasets. The competitors were outperformed by either RACOFS or I-RACOFS in terms of CA and KS measures.

The proposed variations in general have a significant superiority over the competitors, including ant colony and bee colony based algorithms. This superiority was gained as a result of the proportional reliance of the algorithms on the previous traversals of the ants that increases their abilities in exploring and exploiting the solution space. Also, the hybridization of RACOFS with the local improver procedure becomes more significant as the dataset size (i.e. number of features) grows and additionally, poor results for the WI dataset requires that the experiments to be carried out on this dataset with $k>1$.

In Table 7, the comparisons are made for large datasets, HV and LR, in terms of CA and KS. In the HV dataset, the proposed algorithms RACOFS and C-RACOFS have the least significant results, while the integration of RACOFS with the capability hybrid procedure is necessary to enhance the final results. I-RACOFS could improve the solutions with a specific subset size to outperform other algorithms of TSIACO variations and BCOFS. In the ARR dataset, the variations of RACOFS and C-RACOFS were only superior to BCOFS, while showing almost similar performances in comparisons to other algorithms.

The proposed variations use the 1-NN classifier. Although in general satisfactory results were gained in comparison to other algorithms, some of the inferiorities can be alleviated if the proposed algorithms are tested with $k>1$. For example as shown in Fig. 6 (the experiments on the HV dataset), by increasing the number of neighbors, the classifier's accuracy improves. Hence in large datasets it is likely that RACOFS with $k=1$ suffers from the problem of over-fitting even in cross validation rounds and degrade the performance significantly. The significance of hybridization (I-RACOFS) increases as the dataset grows. According to Tables 6 and 7, for $D>30$, I-RACOFS showed better results compared to other variations, in datasets with sizes of $D<30$, in a few cases, improvements were made by I-RACOFS.

In Table 8, the comparisons are only made between datasets that were in common with the PSO algorithm proposed in [5]. In this table, the results are shown in the form of $x(y)$, where x is the result for CA and y is the best subset size. The limited and unlimited expressions of PSO-based algorithms refer to whether the subset sizes are restricted or unrestricted.

Considering the CA measure, superiorities in either of the proposed variations were gained over the PSO-based algorithms, for the SO and GL datasets. Similarly for the KS measure, the proposed variations outperform PSO-based algorithms in the GL and SO datasets. mr^2PSO-based variations could outperform RACOFS-based algorithms only in IO dataset. Although PSO and RACOFS are both swarm-based algorithms, but they have quite different underlying procedures. Furthermore PSO considered here rely on SVM for classification while RACOFS uses k-NN. Hence, the superiorities gained over PSO, in most datasets not only demonstrate the ability of RACOFS and its variations in exploring and

Table 5
Ant and bee colony comparisons on small datasets using CA and KS measures (all the units are in %).

Dataset	d	Proposed algorithms						Recent ACO				Baseline ACO			
		I-RACOFS (l)	I-RACOFS (2)	I-RACOFS (3)	I-RACOFS (4)	RACOFS	C-RACOFS	ACOFS-P	ACOFS-R	TSI ACOl	TSI AC02	AS	MMAS	ACS	BCOFS
MK1	1	50-50	N/A	N/A	N/A	100-93 (3)	100-93 (3)	100-100 (3)	100-100 (3)	100-100 (3)	100-100 (3)	68.45-68. 44 (3)	100-100 (3)	100-100 (3)	50-49.9 (2)
	2	66.6-54.5	66.6-54.5	N/A	N/A										
	4	97.2-93	97.2-93	97.2-93	97.2-93										
	5	88.2-93	88.2-93	88.2-93	88.2-93										
MK2	1	67.3-67	N/A	N/A	N/A	79.4-79.4 (6)	79.4-79.4 (6)	65-68.3 (3)	65-68.3 (3)	67.12-67 (2)	67.12-67 (2)	64. 26-64. 26 (3)	64.26-64.26 (3)	64.26-64.2 6 (3)	67.167 (4)
	2	67.3-67	67.3-67	N/A	N/A										
	4	63.4-60	63.4-60	63.4-60	63.4										
	5	68.7-60.5	68.7-60.5	68.7-60.5	68.7-60.5										
PO	2	64.4-64.2	64.4-64.2	N/A	N/A	64.4-61.25 (2)	64.4-61.25 (2)	39.32-39.32 (2)	37.8-37.65 (3)	61.25-61.25 (2)	42.2-61.25 (2)	39. 32-39. 32 (3)	38.2-38.2 (3)	39.3 2-39.3 2 (3)	68-68 (4)
	4	64.4-64.8	64.4-64.8	64.4-64.8	64.4-64.8										
	6	31.1-31.5	31.1-31.5	31.1-31.5	31.1-31.5										
	7	30-31.5	30-31.5	30-31.5	30-31.5										
BC	2	96.3-99.5	96.3-99.8	N/A	N/A	96.9-96 (4)	96.1-96 (4)	94.7-94.7 (3)	94.7-94.7 (3)	96.85-95.8 (4)	95.7-95 (3)	94-94 (2)	94-94 (2)	92.4-92.4 (1)	85.3-85.4 (5)
	4	96.3-99.7	96.3-99.7	96.3-100	N/A										
	6	96.3-938.7	96.3-99.8	96.3-100	96.3-100										
	8	96.3-99.8	96.3-99.8	96.3-99.8	96.3-99.8										
GL	2	99.5-99.5	99.5-99.6	N/A	N/A	100-99.9 (4)	100-99.9 (4)	99.53-99.86 (4)	99.53-99.53 (4)	100-100 (5)	100-100 (5)	98. 6-98. 6 (6)	54.2-54.2 (2)	99-99 (2)	100-100 (4)
	4	100-99.7	100-99.86	100-100	N/A										
	6	100-99.7	100-99.93	100-100	100-100										
	8	100-99.7	100-99.8	100-99.8	100-99.8										
VW	2	48.3-90.8	48.3-90.8	N/A	N/A	99 7-99.8 (8)	99.7-99.8 (8)	93-98.9 (4)	92.8-98.8 (6)	99.6-99 (7)	91.6-91.2 (4)	100-100 (3)	100-100 (3)	100-100 (3)	99-91 (5)
	4	88.9-98.5	88.9-98.5	88.9-98.5	88.9-98.5										
	6	94.2-99.7	94.7-99.7	94.7-99.75	94.7-99.75										
	8	95.7-99.88	95.7-99.88	95.7-99.94	95.7-99.94										

Table 6
Ant and bee colony comparisons on medium datasets using CA and KS measures (all the units are in %).

		Proposed algorithms						Recent ACO				Baseline ACO			
Dataset	d	I-RACOFS(I)	I-RACOFS(2)	I-RACOFS(3)	I-RACOFS(4)	RAC OFS	C-RAC OFS	ACOFS-P	ACOFS -R	TSIA COl	TSIAC 02	AS	MM AS	ACS	BCO FS
WI	3	78.08-76.8	78.08-76.8	78.08-76.8	N/A	79.21-76.8	79.21-76.8	66.4-64.97	83.15-88.7	94.94-94	95.5-95	91.6-91	92.15-92	86.3-86	95-73.2
	5	79.21-76.6	79.21-76.6	79.21-76.6	79.21-76.6	(6)	(6)	(5)	(3)	(6)	(8)	(11)	(7)	(5)	(8)
	8	79.21-76.6	79.21-76.6	79.21-76.6	79.21-76.6										
	10	79.21-76.6	79.21-76.6	79.21-76.6	79.21-76.6										
Zo	3	78.21-97.6	78.21-98.3	78.21-98.3	N/A	99.3-99.3	99.3-99.3	88-88	81.2-94.6	97-97	97-97	98.85-98.85	98.57-98.5	99.4-99.42	51.5-86
	6	97.02-98.6	97.02-99	97.02-99.3	97.02-99.3	(15)	(15)	(7)	(6)	(6)	(5)	(6)	(7)	(9)	(5)
	10	98.01-99	98.01-99.3	98.01-99.6	98.01-100										
	13	98.01-99.3	98.01-99.3	98.01-99.6	98.01-99.6										
HR	6	78.6 -84.5	79.7-88.1	79.8-88.3	79.6-88.3	80-80.8	79-80.8	59.7-59.7	59.5-60	72.4-72	70.6-70	74.8-74.8	77.15-77.15	80-81.3	65.23-65.1
	12	74.75-83.8	79.4-85.5	80.4-85.5	80.4-85.5	(3)	(3)	(4)	(3)	(10)	(12)	(6)	(6)	(6)	(13)
	18	78.1-79.6	79.4-79.6	79.5-79.6	79.6-79.6										
	24	76.85-69.5	77-69.5	77-69.5	77-69.5										
IO	7	95.15-93.7	95.15-94.6	95.15-95.7	95.15-95.7	94.01-93.14	94.01-93.14	57.4-52.6	90.88-90.85	91.16-91.16	91.16-91.16	88.9-88	85-85	91-91	93.16-93.1
	14	94.01-93.4	94.87-93.4	94.87-94.8	94.87-94.8	(15)	(15)	(12)	(6)	(14)	(16)	(10)	(12)	(13)	(15)
	20	94.01-91.7	91.16-91.4	91.45-91.8	93.73-91.8										
	27	83.47-87.5	83.19-87.5	91.45-87.8	91.45-87.8										
Ss	7	100-100	100-100	100-100	100-100	100-100	100-100	100-100	95.75-100	100-100	100-100	100-100	100-100	100-100	100-100
	14	100-100	100-100	100-100	100-100	(2)	(2)	(3)	(14)	(9)	(10)	(7)	(7)	(7)	(5)
	21	100-100	100-100	100-100	100-100										
	28	100-100	100-100	100-100	100-100										
So	12	93.26-96.8	93.26-96.8	93.75-96.8	93.75-96.8	87.5-91.45	89.42-91.45	58.2-57	84-84.54	86.05-86	89.9-89	89.42-89	70.7-70	88.5-88	91-90
	24	93.26-96.8	94.71-97.6	95.2-97.6	93.26-97.6	(11)	(11)	(15)	(25)	(10)	(31)	(19)	(22)	(29)	(42)
	36	92.31-95.6	94.23-95.6	94.7-95.6	94.7-95.6										
	48	92.31-95.4	93.26-96.1	93.26-96.1	93.26-96.1										

Table 7
Comparisons with other swarm based algorithms in large datasets using KS and CA measures (all the units are in %).

Dataset	d	Proposed ACO						Recent ACO				Baseline ACO			BCO
HV		I-RACOFS(1)	I-RACOFS(2)	I-RACOFS(3)	I-RACOFS(4)	RACOFS	C-RACOFS	TSIACO1	TSIACO2	ACOFS-P	ACOFS-R	AS	MMAS	ACS	BCOFS
ARR	20	56.56-59.3	56.56-59.3	56.73-59.35	56.73-59.35	35-35.4	48.7-48	50.62-50.27	52.4-52.03	57.1-58.5	58.75-59.6	61.45-59.08	60.46-59.08	59.14-59.06	54.6-50.6
	40	52.77-59.2	52.77-59.2	54.91-59.27	54.91-59.27	(10)	(52)	(18)	(40)	(48)	(36)	(29)	(39)	(58)	(62)
	60	60.06-58.64	60.06-58.64	60.7-58.62	60.7-58.62										
	80	57.92-52.39	57.92-52.39	57.92-52.39	57.92-52.39										
	56	62.83-64.85	62.83-64.85	62.38-64.85	62.38-64.85	54.86-55.7	55.03-56.1	57.74-56.11	55.5-56.23	55.97-56.51	55.7-56.43	60.95-56.08	62.27-56.09	60.62-56.08	52.4-56.2
	112	63.05-64.8	63.05-64.8	63.05-64.8	63.05-64.8	(273)	(275)	(241)	(170)	(138)	(130)	(28)	(22)	(23)	(139)
	168	61.5-62.5	61.5-62.5	61.5-62.5	61.5-62.5										
	224	57.3-59.92	57.3-59.92	57.3-59.92	57.3-59.92										

exploiting the solution space, but also that *k*-NN performs better than SVM in this case.

5.4.2. Other comparisons

In this section the proposed algorithms are compared with other feature selection algorithms such as classical and genetic algorithms. Since RACOFS is mainly superior over or similar to C-RACOFS, in this set of experiments C-RACOFS is not included. To make justified comparisons only datasets that are in common were selected. Also the proposed algorithms' settings such as number of iterations and independent executions were set based on what was reported in [10]. In Table 9, the comparisons are made with classical approaches like sequential forward selection (SFS), plus-*l* take away-*r* (PTA) and sequential floating forward selection (SFFS). In PTA *r* is not the ripple factor but is the number of features that should be removed. The results are extracted from [10]. In the GL dataset, there is not much difference in the performance. In the VW dataset, I-RACOFS could not outperform the classical algorithms, while RACOFS had similar performances to the classical algorithms with subset size of eight. For the SO dataset RACOFS could only outperform SFS with $d=12$, while the hybrid procedure has superiorities overall.

In the IO dataset RACOFS outperformed all other competitors, and also in the hybrid procedures superiorities are gained over the competitors. Comparing RACO-based feature selection algorithm to the classical algorithms, the proposed algorithm has been mostly superior over the competitors due to the retention of the knowledge of the previously traversed edges, as a rich source of information that is available for the ants to adjust the pheromone appropriately. In Table 10, comparisons are made between hybrid genetic algorithms and the proposed ant-based algorithms, with the same hybridization procedures. The testing conditions are the same and the selected datasets are divided into two disjoint sets of testing and training according to [10] to have justifiable comparisons.

According to the results in Table 10, the overall performance of I-RACOFS algorithms is not better than HGAs. In the SO dataset, for $d=48$, I-RACOFS is superior to HGA only for *r* with values of 2, 3 and 4. In the IO dataset, for $d=27$ and $r=3$ and $r=4$, the outcomes of the I-RACOFS is similar to those of HGAs, while in the other settings of *r* and *d* variables the proposed algorithms are inferior. In the VW dataset, RACOFS has superiority over HGA for *d* with values of 2, 4 and 6. Hybridization is not effective in this dataset to improve the results. Lastly the proposed algorithms in WI dataset are not better than HGAs. Genetic algorithm is an evolutionary based algorithm, while ant colony relies on a different framework (swarm intelligence). Hence, based on the comparisons the genetic algorithm compared here is in general superior over the proposed swarm-based techniques.

5.5. Timing discussions

In this section we investigate the amount of required running time for the proposed feature selection algorithms. In order to show the execution time differences between the proposed variations, a medium-sized dataset such as WI is suitable. According to Fig. 10, RACOFS is a faster algorithm in comparison to C-RACOFS. As the number of ants increase the algorithms need more execution time. Fig. 10 indicates that for a low number of ants (e.g. 50), the execution time differs between RACOFS and C-RACOFS while by increasing the number of ants this difference increases greatly. Hence within a given iteration the execution time of C-RACOFS is more dependent on the number of ants compared to RACOFS.

In Figs. 11–14, timing analysis of the hybrid algorithm of I-RACOFS is shown. Comparisons are made based on the dependencies that the hybrid algorithm has on the desired subset size

Table 8
Comparisons with PSO algorithms proposed in [5] (all units are in %).

Measure	Dataset	d	Proposed ant colony feature selection algorithms						PSO-based algorithms		
			I-RACOFS(1)	I-RACOFS(2)	I-RACOFS(3)	I-RACOFS(4)	RACOFS	C-RACOFS	Mr^2PSO_{Acc} (limited)	Mr^2PSO_{Acc} (unlimited)	Mr^2PSO_{MI} (unlimited)
Classification accuracy (CA)	SO	12	93.26	93.26	93.75	93.75	87.5	89.42	88.15 ± 1.3	85.67 ± 1.7	84.28 ± 1.3
		24	93.26	94.71	95.2	93.26					
		36	92.31	94.23	94.7	94.7	(11)	(11)	(15)	(15)	
		48	92.31	93.26	93.26	93.26					
	IO	7	95.15	95.15	95.15	95.15	94.01	94.01	94.92 ± 0.4	95.44 ± 0.4	95.44 ± 0.4
		14	94.01	94.87	94.87	94.87					
		20	94.01	91.16	91.45	93.73	(15)	(15)	(6)	(6)	
		27	83.47	83.19	91.45	91.45					
	WI	3	78.08	78.08	78.08	N/A	79.21	79.21	99.72 ± 0.3	99.72 ± 0.3	99.19 ± 0.4
		5	79.21	79.21	79.21	79.21					
		8	79.21	79.21	79.21	79.21	(6)	(6)	(6)	(6)	
		10	79.21	79.21	79.21	79.21					
	GL	2	99.5	99.5	N/A	N/A	100	100	79.77 ± 2.0	80.28 ± 1.9	78.5 ± 3.9
		4	100	100	100	100					
		6	100	100	100	100	(4)	(4)	(5)	(5)	
		8	100	100	100	100					
Kappa Statistics (KS)	SO	12	96.8	96.8	96.8	96.8	91.45	91.45	84.35 ± 1.7	81.17 ± 2.3	79.36 ± 1.9
		24	96.8	97.6	97.6	97.6					
		36	95.6	95.6	95.6	95.6					
		48	95.4	96.1	96.1	96.1					
	IO	7	93.7	94.6	95.7	95.7	93.14	93.14	93.47 ± 0.6	94.27 ± 0.5	94.13 ± 0.6
		14	93.4	93.4	94.8	94.8					
		20	91.7	91.4	91.8	91.8					
		27	87.5	87.5	87.8	87.8					
	WI	3	76.8	76.8	76.8	N/A	76.8	76.8	99.64 ± 2.4	99.64 ± 0.5	98.96 ± 0.5
		5	76.6	76.6	76.6	76.6					
		8	76.45	76.45	76.45	76.45					
		10	76	76	76	76					
	GL	2	99.5	99.6	N/A	N/A	99.9	99.9	74.33 ± 2.4	74.83 ± 2.3	72.87 ± 4.8
		4	99.7	99.86	100	N/A					
		6	99.7	99.93	100	100					
		8	99.7	99.8	99.8	99.8					

Table 9
Comparisons with some other baseline feature selection algorithms (classical) using CA (all units are in %).

Datasets	d	SFS	PTA	SFFS	I-RACOFS(1)	I-RACOFS(2)	I-RACOFS(3)	I-RACOFS(4)	RACOFS
GL	2	99.07	99.07	99.07	99.5	99.5	N/A	N/A	100
	4	100	100	100	100	100	100	100	
	6	100	100	100	100	100	100	100	(4)
	8	100	100	100	100	100	100	100	
VW	2	62.02	62.02	62.02	48.3	48.3	N/A	N/A	99.7
	4	92.63	92.83	92.83	88.9	88.9	88.9	88.9	
	6	98.28	98.79	98.79	94.2	94.7	94.7	94.7	(8)
	8	99.70	99.70	99.70	95.7	95.7	95.7	95.7	
SO	12	87.02	89.42	92.31	93.26	93.26	93.75	93.75	87.5
	24	89.90	90.87	93.75	93.26	94.71	95.2	93.26	
	36	88.46	91.83	93.27	92.31	94.23	94.7	94.7	(11)
	48	91.82	92.31	91.35	92.31	93.26	93.26	93.26	
IO	7	93.45	93.45	93.45	95.15	95.15	95.15	95.15	94.01
	14	90.8	92.59	93.73	94.01	94.87	94.87	94.87	
	20	90.03	92.02	92.88	94.01	91.16	91.45	93.73	(15)
	27	89.17	91.17	90.88	83.47	83.19	91.45	91.45	
WI	3	93.82	93.82	93.82	78.08	78.08	78.08	N/A	79.21
	5	94.38	94.38	94.94	79.21	79.21	79.21	79.21	
	8	95.51	95.51	95.51	79.21	79.21	79.21	79.21	(6)
	10	92.13	92.13	92.7	79.21	79.21	79.21	79.21	

and ripple factors. By increasing the desired subset size, d, and ripple factor, r, the running time grows, as the ripple factor values intensify the search around local optima stronger. Hence the algorithm requires longer execution time. However the desired subset size, d, has more impact on the execution time. For instance, in Fig. 11 where $d=3$, the required time for the algorithms to converge are similar. On the other hand as the required subset size increased to 5, according to Fig. 12, the timing complexity between $r=3$ and 4 increase significantly. Also in Figs. 13 and 14 it can be seen that the required execution time for the algorithm is affected greatly as the desired subset size increases. Therefore the timing complexity is more dependent on subset size rather than ripple factor and for datasets with large numbers of features the algorithm requires a large running time. For further analysis of the timing analysis of improver hybrid procedure interested readers can refer to [10].

Table 10
Comparisons with hybrid genetic algorithms using CA (all units are in %).

Dataset	d	HGA(1)	HGA (2)	HGA(3)	HGA(4)	I-RACOFS(1)	I-RACOFS(2)	I-RACOFS(3)	I-RACOFS(4)	RACOFS
GL	2	99.07	99.07	N/A	N/A	99.5	99.5	N/A	N/A	100
	4	100	100	100	100	100	100	100	100	
	6	100	100	100	100	100	100	100	100	(4)
	8	100	100	100	100	100	100	100	100	
SO	12	93.65	94.71	94.61	94.81	93.26	93.26	93.75	93.75	87.5
	24	95.86	95.96	96.34	96.15	93.26	94.71	95.2	93.26	
	36	95.67	95.82	95.67	95.67	92.31	94.23	94.7	94.7	(11)
	48	92.6	93.17	93.17	93.08	92.31	93.26	93.26	93.26	
IO	7	95.38	95.5	95.56	95.56	95.15	95.15	95.15	95.15	94.01
	14	94.93	95.56	95.21	95.21	94.01	94.87	94.87	94.87	
	20	93.9	94.19	93.73	94.13	94.01	91.16	91.45	93.73	(15)
	27	91.45	91.45	91.45	91.45	83.47	83.19	91.45	91.45	
VW	2	62.02	62.02	N/A	N/A	48.3	48.3	N/A	N/A	99.7
	4	92.83	92.83	92.83	92.83	88.9	88.9	88.9	88.9	
	6	98.79	98.79	98.79	98.79	94.2	94.7	94.7	94.7	(10)
	8	99.7	99.7	99.7	99.7	95.7	95.7	95.7	95.7	
WI	3	93.82	93.82	93.82	N/A	78.08	78.08	78.08	N/A	79.21
	5	95.51	95.51	95.51	95.51	79.21	79.21	79.21	79.21	
	8	95.51	95.51	95.51	95.51	79.21	79.21	79.21	79.21	(6)
	10	92.7	92.7	92.7	92.7	79.21	79.21	79.21	79.21	

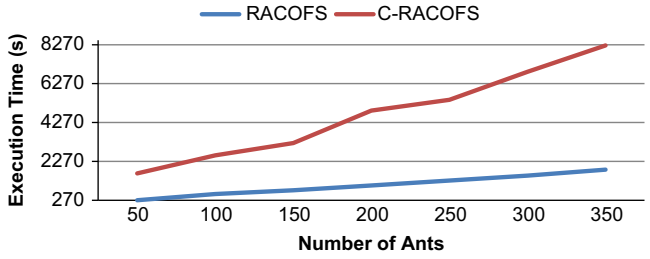

Fig. 10. Comparisons of RACOFS and C-RACOFS.

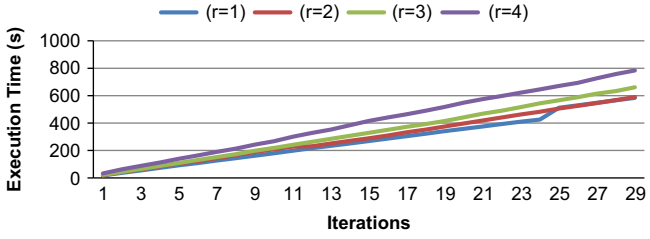

Fig. 11. I-RACOFS timing results for $d=3$.

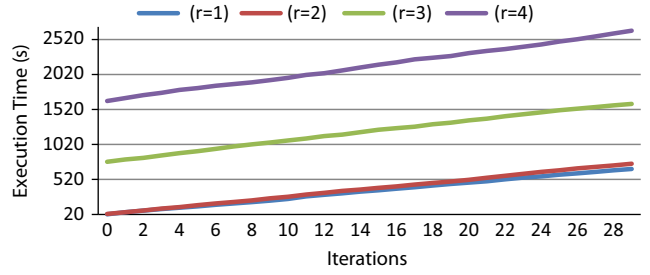

Fig. 12. I-RACOFS timing results for $d=5$.

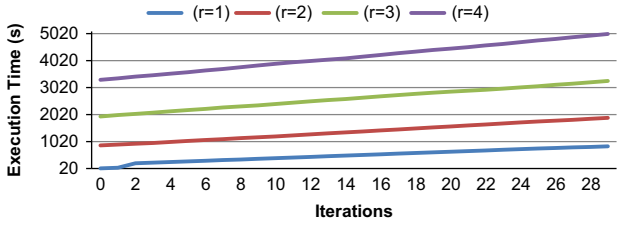

Fig. 13. I-RACOFS timing results for $d=8$.

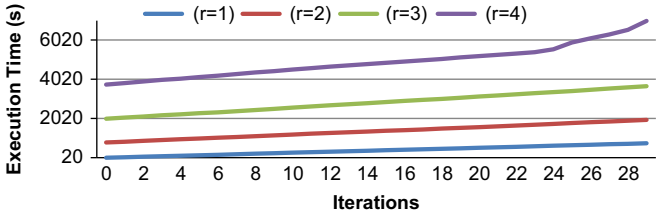

Fig. 14. I-RACOFS timing results for $d=10$.

6. Conclusion and future works

In this paper we introduced a new ant colony algorithm, RACO, that benefits from the traversals of the previously traversed edges. The previous traversals are seen as a rich source of information helping to adjust the pheromone values laid on the edges as accurately as possible. The aim of using previously traversed edges is to provide a new methodology to increase the exploration and exploitation abilities of the ants and correspondingly prevent the algorithm converging prematurely.

Then, RACO is applied to the task of feature selection (RACOFS) to show the effectiveness of the algorithm in its application. It was assumed that RACOFS suffers from the problem of inequality of selection. Hence C-RACOFS, a second variation, was introduced to test this assumption. RACOFS is capable of finding globally optimal solutions, but is prone to be entrapped in local optimal. Therefore the third variation, I-RACOFS, was introduced that integrates a local search procedure with RACOFS and investigates further possible improvements by searching the vicinity of the globally optimal solution.

RACOFS, compared to other ant-based feature selection algorithms, showed significant superiority in both the KS and CA measures. This superiority is gained as a result of reliance on the previous iterations' traversals. RACOFS is capable of using the rich sources of previously traversed edges and finding globally optimal

solutions, in small and medium datasets, while in large datasets hybrid algorithms are better to reach an optimal solution. In order to reach an optimal solution in large datasets, RACOFS is required to be executed with k-NN where $k > 1$. The timing analysis results indicate RACOFS as the fastest variation in comparison to two other variations, and I-RACOFS is the slowest variation. The execution time of I-RACOFS heavily depends on the hybridization parameters of ripple factor and the desired subset size, while as the experiments indicated the execution time is more dependent on the desired subset size, rather than ripple factor.

As some suggestions for future work, although the proposed algorithm showed significant performances using k-NN with $k=1$, the proposed algorithms can be extended by testing for different values of k (e.g. $k > 1$), especially for the WI dataset, to investigate further improvements for this dataset. The hybridization is applied to the outcome of RACOFS while it also can be applied to C-RACOFS or to each solution created by the ants.

Acknowledgement

We would like to thank Dr. Myra Wilson, Aberystwyth University, for reviewing early drafts of this paper.

References

[1] R. Jensen, Q. Shen, Fuzzy-rough attribute reduction with application to web categorization, Fuzzy Sets Syst. 141 (3) (2004) 469–485.
[2] Q. Shen, R. Jensen, Selecting informative features with fuzzy-rough sets and its application for complex systems monitoring, Pattern Recognit. 37 (7) (2004) 1351–1363.
[3] M.D. Monirul Kabir, M.D. Monirul Islam, K. Murase, A new hybrid ant colony optimization algorithm for feature selection, Expert Syst Appl 39 (2012) 3747–3763.
[4] R. Forsati, A. Moayedikia, B. Safarkhani, Heuristic approach to solve feature selection problem, Lecture Notes in Computer Science, Verlag Berlin Heidelberg, Springer (2011) 707–717.
[5] A. Unler, A. Murat, R.B. Chinnam, mr2PSO: a maximum relevance minimum redundancy feature selection method based on swarm intelligence for support vector machine classification, Inf. Sci. 181 (2011) 4625–4641.
[6] M. Dorigo, V. Maniezzi, A. Colony, Ant system: optimization by a colony of cooperating agents, IEEE Trans. Syst. Man Cybern. Part B Cybern. 26 (1) (1996) 29–41.
[7] T. Stutzle, H.H. Hoos, Max–min ant system, Future Gener. Comput. Syst. 16 (8) (2000) 889–914.
[8] L.M. Gambardella, M. Dorigo, Solving symmetric and asymmetric TSPs by ant colonies, in: In Evolutionary Computation, 1996, Proceedings of IEEE International Conference on, 1996, pp. 622–627.
[9] H.H. Hsu, C.W. Hsieh, M.D. Lu, Hybrid feature selection by combining filters and wrappers, Expert Syst Appl 38 (7) (2011) 8144–8150.
[10] Il-S. Oh, J-S. Lee, B.-R. Moon, Hybrid genetic algorithms for feature selection, IEEE Trans. Pattern Anal. Mach. Intell. 26 (11) (2004) 1424–1437.
[11] C.L. Huang, ACO-based hybrid classification system with feature subset selection and model parameters optimization, Neurocomputing 73 (2009) 438–448.
[12] R. Jensen, Q. Shen, Fuzzy-rough data reduction with ant colony optimization, Fuzzy Sets Syst. 149 (2005) 5–20.
[13] R. Forsati, A. Moayedikia, A. Keikhah, A novel approach for feature selection based on the bee colony optimization, Int. J. Comput. Appl. Technol. 43 (8) (2012) 30–34.
[14] C. Jun Tan, C. Peng Lim, Y.N. Cheah, A multi-objective evolutionary algorithm-based ensemble optimizer for feature selection and classification with neural network models, Neurocomputing 125 (11) (2014) 217–228.
[15] R. Diao, Q. Shen, Two new approaches to feature selection with harmony search, in: Fuzzy Systems (FUZZ), 2010 IEEE International Conference on, Barcelona, Spain, 2010.
[16] M. Mirkhani, R. Forsati, Alireza Mohammad Shahri, Alireza Moayedikia, A novel efficient algorithm for mobile robot localization, Rob. Autom. Syst. (2013).
[17] R. Forsati, M. Mahdavi, Web Text Mining Using Harmony Search, Springer, Berlin Heidelberg (2010) 51–64.
[18] M.D. Monirul Kabir, M. Shahjahan, K. Murase, A new local search based hybrid genetic algorithm for feature selection, Neurocomputing 74 (2011) 2914–2928.
[19] S.M. Vieira, J.M.C. Sousa, T.A. Runkler, Two cooperative ant colonies for feature selection using fuzzy models, Expert Syst Appl 37 (2010) 2714–2723.
[20] L. Ke, Z. Feng, Z. Ren, An efficient ant colony optimization approach to attribute reduction in rough set theory, Pattern Recognit. Lett. 29 (9) (2008) 1351–1357.
[21] Y. Chen, D. Miao, R. Wang, A rough set approach to feature selection based on ant colony optimization, Pattern Recognit. Lett. 31 (3) (2010) 226–233.
[22] X. Wang, J. Yang, X. Teng, W. Xaia, R. Jensen, Feature selection based on rough sets and particle swarm optimization, Pattern Recognit. Lett. 28 (2007) 459–471.
[23] L.Y. Chuang, S.W. Tsai, C.H. Yang, Improved binary particle swarm optimization using catfish effect for feature selection, Expert Syst. Appl. 38 (10) (2011) 12699–12707.
[24] C.L. Huang, J.F. Dun, A distributed PSO–SVM hybrid system with feature selection and parameter optimization, Appl. Soft Comput. 8 (2008) 1381–1391.
[25] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Neural Networks, 1995. Proceedings, IEEE International Conference on, 1995, pp. 1942–1948.
[26] P. Ravisankar, V. Ravi, G. Raghava Rao, I. Bose, Detection of financial statement fraud and feature selection using data mining techniques, Decis. Support. Syst. 50 (2) (2011) 491–500.
[27] C.F. Tsai, Y.C. Hsiao, Combining multiple feature selection methods for stock prediction: union, intersection, and multi-intersection approaches, Decis. Support. Syst. 50 (1) (2010) 258–269.
[28] Y. Chen, D. Liginlal, A maximum entropy approach to feature selection in knowledge-based authentication, Decis. Support. Syst. 46 (1) (2008) 388–398.
[29] A. Duric, F. Song, Feature selection for sentiment analysis based on content and syntax models, Decis. Support. Syst. 53 (4) (2012) 704–711.
[30] D. Chyzhyk, A. Savio, M. Grana, Evolutionary ELM wrapper feature selection for Alzheimer's disease CAD on anatomical brain MRI, Neurocomputing November (2013).
[31] M. Dorigo, G.D. Caro, L.M. Gambradella, Ant algorithms for discrete optimization, Artif. Life 5 (2) (1999) 137–172.
[32] Z. Zhang, Z. Feng, Two-stage updating pheromone for invariant ant colony optimization algorithm, Expert Syst. Appl. 39 (2012) 706–712.
[33] J. Cohen, A coefficient of agreement for nominal scales, Educ. Psychol. Meas. 20 (1) (1960) 37–46.
[34] A. Tsymbal, M. Pechenizkiy, P. Cunningham, Diversity in search strategies for ensemble feature selection, Inf. Fusion 6 (1) (2005) 83–98.
[35] J. Huang, Y. Cai, X. Xu, A hybrid genetic algorithm for feature selection wrapper based on mutual information, Pattern Recognit. Lett. 28 (2007) 1825–1844.
[36] A.M. Fraster, H.L. Swinney, Independent coordinates for strange attractors from mutual information, Phys. Rev. A: At. Mol. Opt. Phys. 33 (2) (1986) 1134–1140.

Rana Forsati is a Ph.D. student at Shahid Beheshti University, Tehren, Iran. She is the member of NLP Research Laboratory of Electrical and Computer Engineering faculty. She is a visiting scholar at University of Minnesota at the Department of Computer Engineering. Her research interests include machine learning, soft computing and data mining with applications in natural language processing and recommender systems.

Alireza Moayedikia received the B.S. degree in Computer Engineering from Islamic Azad University of Karaj, Karaj, Iran in 2011. In late 2013 he graduated with M.Sc. of software engineering of Staffordshire University, U.K. He started working as a research assistant at Shahid Beheshti University, since 2011. He has published few papers on evolutionary and swarm intelligence topics in different areas of computer science such as robotics and feature selection. His research interests are Data/Text Mining, Evolutionary and Swarm based Algorithms, Machine Learning and Recommender Systems.

Richard Jensen received the B.Sc. degree in computer science from Lancaster University, U.K, and the M.Sc. and Ph.D. degrees in artificial intelligence from the University of Edinburgh, U.K. He is a Lecturer with the Department of Computer Science at Aberystwyth University. His research interests include rough and fuzzy set theory; pattern recognition; information retrieval; feature selection; and swarm intelligence. He has published over 60 peer-refereed articles in these areas. He is on the editorial board of Transactions on Rough Sets amongst others, and on the advisory board of the International Rough Set Society.

Mehrnoush Shamsfard has received her BS and MS both on computer software engineering from Sharif University of Technology, Tehran, Iran. She received her PhD in Computer Engineering- Artificial Intelligence from Amir Kabir University of Technology in 2003. Dr. Shamsfard has been an assistant professor at Shahid Beheshti University from 2004. She is the head of NLP research Laboratory of Electrical and Computer Engineering faculty. Her main fields of interest are natural language processing, ontology engineering, text mining and semantic web.

Mohammad Reza Meybodi received his B.S. and M.S. degrees in Economics from Shahid Beheshti University in Iran during 1973 and 1977, respectively. Then he received his M.S. and Ph.D. degrees from Oklahoma University, USA during 1980 and 1983, respectively in Computer Science. Presently, he is a Full Professor in Computer Engineering Department, Amirkabir University of Technology, Tehran, Iran. Prior to the present position, he worked from 1983 to 1985 at Western Michigan University and from 1985 to 1991 at Ohio University, USA, as an Assistant Professor. Some of his research interests include learning systems, parallel algorithms, soft computing.