Expert Systems with Applications 39 (2012) 3747–3763

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

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

A new hybrid ant colony optimization algorithm for feature selection

Md. Monirul Kabir [a], Md. Shahjahan [b], Kazuyuki Murase [a,c,*]

[a] Department of System Design Engineering, University of Fukui, Fukui, Japan
[b] Department of Electrical and Electronic Engineering, Khulna University of Engineering and Technology (KUET), Khulna, Bangladesh
[c] Department of Human and Artificial Intelligence Systems and Research and Education Program for Life Science, University of Fukui, Fukui, Japan

ARTICLE INFO

Keywords:
Feature selection
Ant colony optimization
Wrapper and filter approaches
Hybrid search
Neural network
Classification accuracy

ABSTRACT

In this paper, we propose a new hybrid ant colony optimization (ACO) algorithm for feature selection (FS), called ACOFS, using a neural network. A key aspect of this algorithm is the selection of a subset of salient features of reduced size. ACOFS uses a hybrid search technique that combines the advantages of wrapper and filter approaches. In order to facilitate such a hybrid search, we designed new sets of rules for pheromone update and heuristic information measurement. On the other hand, the ants are guided in correct directions while constructing graph (subset) paths using a bounded scheme in each and every step in the algorithm. The above combinations ultimately not only provide an effective balance between exploration and exploitation of ants in the search, but also intensify the global search capability of ACO for a high-quality solution in FS. We evaluate the performance of ACOFS on eight benchmark classification datasets and one gene expression dataset, which have dimensions varying from 9 to 2000. Extensive experiments were conducted to ascertain how AOCFS works in FS tasks. We also compared the performance of ACOFS with the results obtained from seven existing well-known FS algorithms. The comparison details show that ACOFS has a remarkable ability to generate reduced-size subsets of salient features while yielding significant classification accuracy.

1. Introduction

Feature selection (FS) is generally used in machine learning, especially when the learning task involves high-dimensional datasets. High-dimensional datasets contain very large feature sets, which not only cause learning to be more difficult, but also degrade the generalization performance of the learned models. The purpose of FS is to simplify and enhance the quality of a dataset by selecting salient features. Ordinarily, FS deletes spurious features from the original dataset without sacrificing generalization performance. In real-world problems, FS is essential due to the existence of the following factors: (a) abundance of noise, (b) spurious information, and (c) irrelevant and redundant features in the original feature set. Accordingly, FS has become an area of active research spreading throughout many fields, including pattern recognition, data mining, image mining, and text categorization (Aghdam, Aghaee, & Basiri, 2009; Jensen, 2005).

A number of proposed approaches for FS can broadly be categorized into the following three classifications: wrapper, filter, and hybrid (Liu & Tu, 2004). In the wrapper approach, a predetermined learning model is assumed, wherein features are selected that justify the learning performance of the particular learning model (Guyon & Elisseeff, 2003), whereas in the filter approach, statistical analysis of the feature set is required, without utilizing any learning model (Dash & Liu, 1997). Schematic diagrams of how the wrapper and filter approaches find salient features are given in Fig. 1. The hybrid approach attempts to utilize the complementary strengths of the wrapper and filter approaches (Huang, Cai, & Xu, 2007).

Subsets can be generated and the search process carried out in a number of ways. One method, called sequential forward search (SFS; Guan, Liu, & Qi, 2004; Peng, Long, & Ding, 2003), is to start the search process with an empty set and successfully add features; another option called sequential backward search (SBS; Gasca, Sanchez, & Alonso, 2006; Hsu, Huang, & Schuschel, 2002), is to start with a full set and successfully remove features. In addition, a third alternative, called bidirectional selection (Caruana & Freitag, 1994), is to start on both ends and add and remove features simultaneously. A fourth approach (Lai, Reinders, & Wessels, 2006; Straceezzi & Utgoff, 2004), is to have a search process start with a randomly selected subset using a sequential or bidirectional strategy. Yet another search strategy, called complete search (Liu & Tu, 2004), may give a best solution to an FS task due to the thoroughness of its search, but is not feasible when dealing with a large number of features. For example, assuming s to be a subset of

* Corresponding author at: Department of Human and Artificial Intelligence Systems, University of Fukui, 3-9-1 Bunkyo, Fukui 910-8507, Japan. Tel.: +81 (0) 776 27 8774; fax: +81 (0) 776 27 8420.
E-mail address: murase@synapse.his.u-fukui.ac.jp (K. Murase).

0957-4174/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2011.09.073

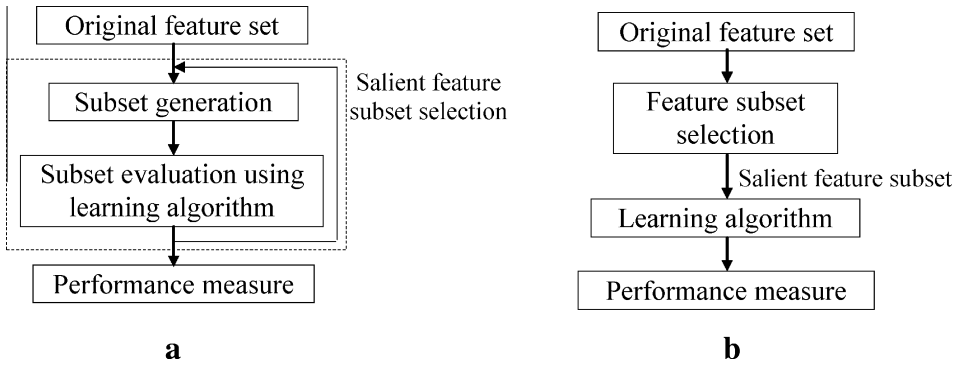

Fig. 1. Schematic diagrams of (a) wrapper approach and (b) filter approach.

selected features, and n to be the number of available features, the total computational cost of the combination is ${}^{n}c_{s} \approx n!/(s! \cdot (n-s)!)$. The worst case scenario occurs when the values of n and s become large. Alternatively, the sequential strategy is simple to implement and fast, but is affected by the nesting effect (Pudil, Novovicova, & Kittler, 1994), wherein once a feature is added (or, deleted) it cannot be deleted (or, added) later. In order to overcome such disadvantages of the sequential search strategy, another search strategy, called the floating search strategy (Pudil et al., 1994), has been implemented.

Most of the afore-mentioned search strategies, however, attempt to find solutions in FS that range between sub-optimal and near optimal regions, since they use local search throughout the entire process, instead of global search. On the other hand, these search algorithms utilize a partial search over the feature space, and suffer from computational complexity. Consequently, near-optimal to optimal solutions are quiet difficult to achieve using these algorithms. As a result, many research studies now focus on global search algorithms (or, "metaheuristics") (Ke, Feng, & Ren, 2008). The significance of global search algorithms is that they can find a solution in the full search space on the basis of activities of multi-agent systems that use a global search ability utilizing local search appropriately, thus significantly increasing the ability of finding very high-quality solutions within a reasonable period of time (Dorigo & Stutzle, 2004). To achieve global search, researchers have attempted simulated annealing (Filippone, Masulli, & Rovetta, 2006), genetic algorithm (Yang & Honavar, 1998), ant colony optimization (Aghdam et al., 2009; Ani, 2005), and particle swarm optimization (Wang, Yang, Teng, Xia, & Jensen, 2007) algorithms in solving FS tasks.

In this paper, we propose a new hybrid ant colony optimization (ACO)-based FS algorithm (ACOFS) that utilizes a hybrid search strategy in the feature space. The idea incorporated in this algorithm was originally introduced in our earlier work (Kabir, Shahjahan, & Murase, 2009). The main focus of this algorithm is to generate subsets of salient features of reduced size. The proposed method utilizes a hybrid search technique that combines the wrapper and filter approaches. It has often been found that hybrid techniques are capable of finding a good solution, even when a single technique is often trapped with an incomplete solution. In this regard, ACOFS modifies the standard pheromone update and heuristic information measurement rules based on the above two approaches. The reason for the novelty and distinctness of the proposed ACOFS *versus* previous algorithms (Aghdam et al., 2009; Ani, 2005; Kanan, Faez, & Taheri, 2007; Ke et al., 2008; Khushaba, Alsukker, Ani, & Jumaily, 2008; Robbins, Zhang, & Bertrand, 2008; Sivagaminathan & Ramakrishnan, 2007) lie in the following two aspects.

First, ACOFS emphasizes not only the selection of a number of salient features, but also the attainment of a reduced number of them. ACOFS selects salient features of a reduced number using a subset size determination scheme. Such a scheme works upon a bounded region and provides sizes of constructed subsets that are smaller in number. Thus, following this scheme, an ant attempts to traverse the node (or, feature) space to construct a path (or, subset). This approach is quite different from those of the existing schemes (Aghdam et al., 2009; Kanan et al., 2007; Ke et al., 2008), where the ants are guided by using the SFS strategy in selecting features during the feature subset construction (SC). However, a problem is that, SFS requires an appropriate stopping criterion to stop the SC. Otherwise, a number of irrelevant features may be included in the constructed subsets, and the solutions may not be effective. To solve this problem, some algorithms (Ani, 2005; Khushaba et al., 2008; Robbins et al., 2008; Sivagaminathan & Ramakrishnan, 2007) define the size of a constructed subset by a fixed number for each iteration for all ants, which is incremented at a fixed rate for following iterations. This technique could be inefficient if the fixed number becomes too large or too small. Therefore, deciding the subset size within a reduced area may be a good step for constructing the subset while the ants traverse through the feature space.

Second, ACOFS utilizes a hybrid search technique for selecting salient features that combines the advantages of the wrapper and filter approaches. An alternative name for such a search technique is "ACO search". This technique is designed with two sets of new rules for pheromone update and heuristic information measurement. The idea of these rules is based mainly on the random and probabilistic behaviors of ants while selecting features during SC. The aim is to provide the correct information to the features and to maintain an effective balance between exploitation and exploration of ants during SC. Thus, ACOFS achieves a strong search capability that helps to select a smaller number of the most salient features among a feature set. In contrast, the existing algorithms (Aghdam et al., 2009; Ani, 2005; Kanan et al., 2007; Ke et al., 2008; Khushaba et al., 2008; Robbins et al., 2008; Sivagaminathan & Ramakrishnan, 2007) try to design rules without distinguishing between the random and probabilistic behaviors of ants during the construction of a subset. Consequently, ants may be deprived of the opportunity of utilizing enough previous experience or investigating more salient features during SC in their solutions.

The rest of this paper is organized as follows. Section 2 describes some related works concerning FS. A detailed description of our proposed ACOFS, including the computational complexity of different stages, is described in detail in Section 3. Section 4 presents the results of our experimental studies, including the methodology, results, and a comparison with other existing FS

algorithms. Finally, Section 5 concludes the paper with a brief summary and a few remarks.

2. Previous work

Search strategy considerations for any FS algorithm are a vital part in finding salient features of a given dataset (Liu & Tu, 2004). Numerous algorithms have been proposed to address the problem of searching. Most algorithms use either a sequential search (for example, Abe, 2005; Gasca et al., 2006; Guan et al., 2004; Hsu et al., 2002; Setiono & Liu, 1997) or a global search (e.g., Huang et al., 2007; Ke et al., 2008; Muni, Pal, & Das, 2006; Oh, Lee, & Moon, 2004; Robbins et al., 2008; Wang et al., 2007; Yang & Honavar, 1998). On the basis of guiding the search strategies and evaluating the subsets, in contrast, the existing FS algorithms can be grouped into the following three approaches: wrapper (e.g., Abe, 2005; Gasca et al., 2006; Pal & Chintalapudi, 1997; Rakotomamonjy, 2003; Verikas & Bacauskiene, 2002; Wang, Zhou, & Chu, 2008), filter (e.g., Chow & Huang, 2005; Hall, 2000; Sindhwani et al., 2004), and hybrid (e.g., Huang et al., 2007; Sivagaminathan & Ramakrishnan, 2007). It is well-known that wrapper approaches always return features with a higher saliency than filter approaches, as the former utilize the association of features collectively during the learning process, but are computationally more expensive (Liu & Tu, 2004). In addition, there are several works in which sufficient surveys regarding FS are discussed (see Guyon & Elisseeff, 2003). The following subsections outline the FS process based on the above search algorithms, in order that one may visualize the ideas and shortcomings behind different techniques together with the position of ACOFS among them.

In solutions for FS, filter approaches are faster to implement, since they estimate the performance of features without any actual model assumed between outputs and inputs of the data. A feature can be selected or deleted on the basis of some predefined criteria, such as, mutual information (Chow & Huang, 2005), principal component analysis (Kambhatla & Leen, 1997), independent component analysis (Back & Trappenberg, 2001), class separability measure (Mao, 2002), or variable ranking (Caruana & Sa, 2003). Filter approaches have the advantage of computational efficiency, but the saliency of the selected features is insufficient, because they do not take into account the biases of classification models.

In order to implement the wrapper approaches, a number of algorithms (Abe, 2005; Gasca et al., 2006; Guan et al., 2004; Hsu et al., 2002; Kabir, Islam, & Murase, 2008; Kabir, Islam, & Murase, 2010) have been proposed that use sequential search strategies in finding a subset of salient features. In Guan et al. (2004), features are added to a neural network (NN) according to SFS during training. The addition process is terminated when the performance of the trained classifier is degraded. Recently proposed approaches (e.g., Kabir et al., 2008; Kabir et al., 2010) have drawn much attention in SFS-based FSs. In these approaches, correlated (distinct) features from two groups, namely, similar and dissimilar, are added to the NN training model sequentially. At the end of the training process, when the NN classifier has captured all the necessary information of a given dataset, a subset of salient features is generated with reduced redundancy of information. In a number of other studies (e.g., Abe, 2005; Gasca et al., 2006; Hsu et al., 2002), SBS is incorporated in FS using a NN where the least salient features have been deleted in stepwise fashion during training. In this context, different algorithms employ different heuristic techniques for measuring saliency of features. In Guan et al. (2004), saliency of features is measured using a NN training scheme in which only one feature is used in the input layer at a time. Two different weight analysis-based heuristic techniques are employed in Gasca et al. (2006) and Hsu et al. (2002) for computing the saliency of features. Furthermore, in Abe (2005), a full feature set NN training scheme is used where each feature is temporarily deleted with a cross-check of NN performance.

The value of a loss function, consisting of cross entropy with a penalty function, is considered directly for measuring the saliency of a feature in Setiono and Liu (1997) and Verikas and Bacauskiene (2002). In Setiono and Liu (1997), the penalty function encourages small weights to converge to zero, or prevents weights from converging to large values. After the penalty function has finished running, those features that have smaller weights are sequentially deleted during training as being irrelevant. On the other hand, in Verikas and Bacauskiene (2002), the penalty function forces a network to keep the derivatives of the values of its neurons' transfer functions low. The aim of such a restriction is to reduce output sensitivity to input changes. In the FS process, feature removal operations are performed sequentially, especially for those features that do not degrade accuracy of the NN upon removal. A class-dependent FS algorithm in Wang et al. (2008), selects a desirable feature subset for each class. It first divides a C class classification problem into C two-class classification problems. Then, the features are integrated to train a support vector machine (SVM) using a SFS strategy in order to find the feature subset of each binary classification problem. Pal and Chintalapudi (1997) has proposed a SBS-based FS technique that multiplies an attenuation function by each feature before allowing the features to be entered into the NN training. This FS technique is the root for proposing another FS algorithm in Chakraborty and Pal (2004). Rakotomamonjy (2003) has proposed new FS criteria that are derived from SVMs and that are based on the sensitivity of generalization error bounds with respect to features.

Unlike sequential search-based FS approaches, global search approaches (or, "meta-heuristics") start a search in a full feature space instead of a partial feature space in order to find a high-quality solution. The strategy of these algorithms is based on the mutual cooperation of individual agents. A standard genetic algorithm (GA) has been used for FS (Yang & Honavar, 1998) where fixed length strings in a population set represent a feature subset. The population set evolves over time to converge to an optimal solution via crossover and mutation operations. A number of other algorithms exist (Dash & Liu, 1997; Huang et al., 2007) in which GAs are used for solving FS. A hybrid approach (Huang et al., 2007) for FS has been proposed that incorporates the filter and wrapper approaches in a cooperative manner. A filter approach involving mutual information computation is used here as a local search to rank features. A wrapper approach involving GAs is used here as global search to find a subset of salient features from the ranked features. In Dash and Liu (1997), two basic operations, namely, deletion and addition are incorporated that seek the least significant and most significant features for making a stronger local search during FS.

ACO is predominantly a useful tool, considered as a modern algorithm that has been used in several studies (Aghdam et al., 2009; Ani, 2005; Kanan et al., 2007; Ke et al., 2008; Khushaba et al., 2008; Robbins et al., 2008; Sivagaminathan & Ramakrishnan, 2007) for selecting salient features. During the operation of this algorithm, a number of artificial ants traverse the feature space to construct feature subsets iteratively. During SC, the existing approaches (Aghdam et al., 2009; Ani, 2005; Kanan et al., 2007; Khushaba et al., 2008; Robbins et al., 2008; Sivagaminathan & Ramakrishnan, 2007) define the size of the constructed subsets by a fixed number for each iteration, whereas the SFS strategy has been followed in Aghdam et al. (2009), Kanan et al. (2007) and Ke et al. (2008). In order to measure the heuristic values of features during FS, some of the algorithms (Ani, 2005; Ke et al., 2008; Khushaba et al., 2008; Robbins et al., 2008) use filter tools. Evaluating the constructed subsets is, on the other hand, a vital part in

the study of ACO-based FS, since most algorithms design the pheromone update rules on the basis of outcomes of subset evaluations. In this regard, a scheme of training classifiers (i.e., wrapper tools) has been used in almost all of the above ACO-based FS algorithms, except for the two cases where rough set theory and the latent variable model (i.e., filter tools) are considered, which are in Ke et al. (2008) and Robbins et al. (2008), respectively.

A recently proposed (Wang et al., 2007) FS approach is based on rough sets and a particle swarm optimization (PSO) algorithm. A PSO algorithm is used for finding a subset of salient features over a large and complex feature space. The main heuristic strategy of PSO in FS is that particles fly up to a certain velocity through the feature space. PSO finds an optimal solution through the interaction of individuals in the population. Thus, PSO finds the best solution in the FS as the particles fly within the subset space. This approach is more efficient than a GA in the sense that it does not require crossover and mutation operators; simple mathematical operators are required only.

Most of the global search approaches discussed above do not use a bounded scheme to decide the size of the constructed subsets. Accordingly, in these algorithms, the selected subsets might be larger in size and include a number of least significant features. Furthermore, most of the ACO-based FS approaches do not consider the random and probabilistic behavior of ants during SCs. Thus, the solutions found in these approaches might be incomplete in nature. On the other hand, the above sequential search-based FS approaches suffer from the nesting effect as they try to find subsets of salient features using a sequential search strategy. It is said that such an effect affects the generalization performance of the learning model (Pudil et al., 1994).

3. The proposed ACOFS

In order to avoid the problem of determining the sizes of the subsets, ACOFS uses a bounded scheme, that is to say, a subset size determination scheme. This scheme guides the ants to construct subsets in a reduced form. The approach used by the ants in constructing individual subsets during SC can be seen in Fig. 2. However, it should be kept in mind that the restriction upon the subset size determination is not an inherent constraint. Because, it can be observed that after a certain range, the extended boundary for the bounded scheme results in ineffective solutions for FS. In order to solve another problem, that is to say, incomplete solutions to ACO-based FS algorithms; our ACOFS incorporates a hybrid search strategy (i.e., a combination of the wrapper and filter approaches) by designing different rules to strengthen the global search ability of the ants. Incorporation of these two approaches results in an ACOFS that achieves high-quality solutions for FS from a given dataset.

The steps of ACOFS can be described by the flowchart shown in Fig. 3, which are described in more details as follows.

Step (1) Let N be a feature set of a given dataset D consisting d distinct classes, C_c (for $c = 1,2,\ldots,d$). Suppose n is the total number of features of N. Initialize the pheromone trails τ and the heuristic information η of all n features by assigning equal values to τ and η.

Step (2) Measure the information gain of individual n features using information gain measurement scheme. In this work, we use this information gain property of features as a filter tool in designing rules in order to provide hybrid search throughout the FS process.

Step (3) Generate a set of artificial k ants equivalent to n, that is, $k = n$.

Step (4) Decide the subset size r prior to SC for each of the k ants according to the subset size determination scheme. After that, follow the conventional probabilistic transition rule (Ke et al., 2008) for selecting features with which to construct the subsets as follows:

$$P_i^k(t) = \begin{cases} \frac{[\tau_i(t)]^\alpha [\eta_i(t)]^\beta}{\sum_{u \in j^k} [\tau_u(t)]^\alpha [\eta_u(t)]^\beta} & \text{if } i \in j^k \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

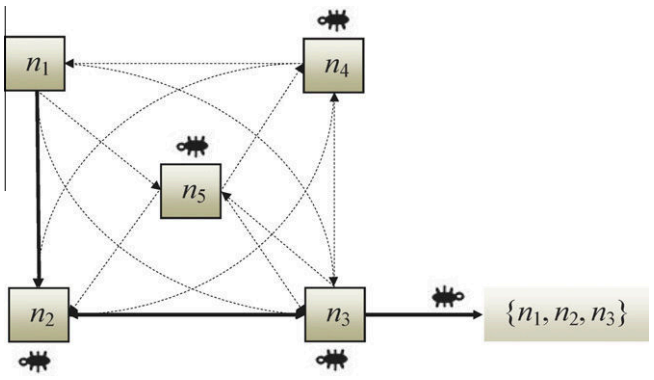

Fig. 2. Representation of subset constructions by individual ants in ACO algorithm for FS. Here, $n_1, n_2, \ldots, n_5$ represent the individual features. As an example, one ant placed in n_1 constructed one subset $\{n_1, n_2, n_3\}$.

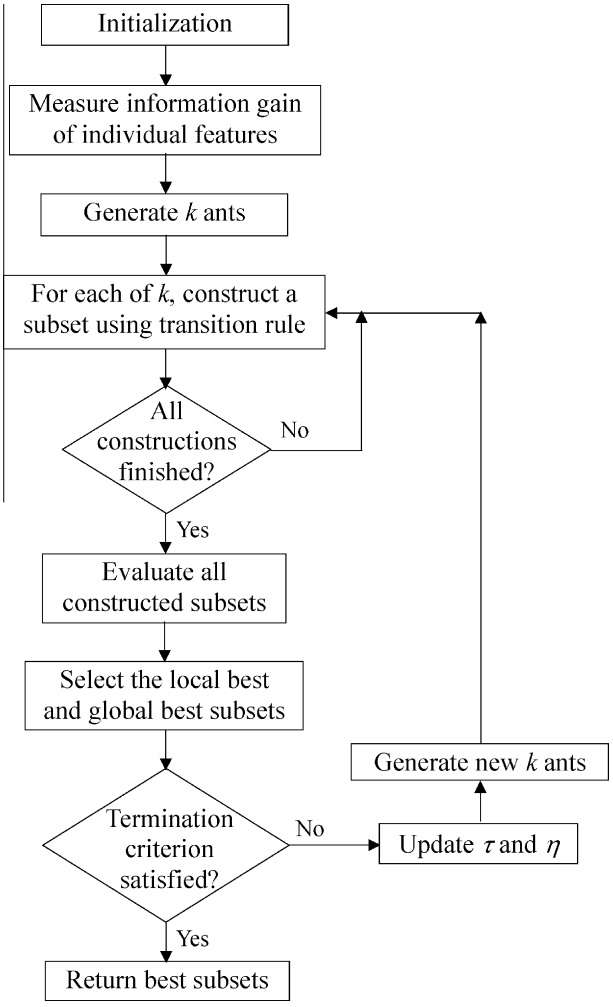

Fig. 3. Flowchart of ACOFS.

where j^k is the set of feasible features that can be added to the partial solution, τ_i and η_i are the pheromone and heuristic values associated with feature i ($i = 1,2,\ldots,n$), and α and β are two parameters that determine the relative importance of the pheromone value and heuristic information. Note that, since the initial value of τ and η for all individual features are equal, Eq. (1) shows random behavior in SC initially.

Step (5) Check the progress of the construction to determine whether it has completed. If the construction of subsets has been completed by all the ants, then continue; otherwise, proceed to Step (4).

Step (6) Evaluate the subsets $S^k(t)$ according to the subset evaluation scheme and measure the classification performance $\gamma(S^k(t))$. Here, $S^k(t)$ refers to the subsets constructed by k ants at iteration t.

Step (7) Select the local best subset, $S^l(t)$ among all $S^k(t)$ and the global best subset, S^g among all $S^l(t)$ in accordance with the selection scheme. Here, $t = 1,2,3,\ldots,I$ where I is a number of iterations.

Step (8) Check whether $S^l(t)$ achieves a predefined accuracy, or the algorithm executes a iteration threshold, I_{th}, then terminate the FS process. Precisely, I_{th} refers to a certain number of iterations in which the algorithm cannot find out any more change of S^g. However, if the termination criterion is satisfied, then S^g is, therefore, retained as a solution of best subset; otherwise, continue. Store the performances of all local best subsets, $\gamma(S^l(t))$ for further use.

Step (9) Update the values of τ and η for all features according to the rules of pheromone update and heuristic information measurement, respectively.

Step (10) Generate a new set of artificial k ants and proceed to through the procedures similarly.

It is now clear that the idea behind ACOFS is straightforward, i.e., guiding the ants using a bounded scheme and providing hybrid technique to the ants' search. To achieve an effective hybrid search, an information gain measurement procedure has been integrated that does not require expensive computation and performed only once throughout the FS process. For better understanding, details about each aspect of ACOFS are now given in the following sections.

3.1. Determination of subset size

In an ACO algorithm, the activities of ants have significance for solving different combinatorial optimization problems. Therefore, in solving the FS problem, guiding ants in the correct directions is very advantageous in this sense. In contrast to other existing ACO-based FS algorithms, our ACOFS uses a straightforward mechanism to determine the subset size r. It employs a simpler probabilistic formula with a constraint and a random function. The aim of using such a probabilistic formula is to provide information to the random function in such a way that the minimum subset size has a higher probability of being selected. This is important in the sense that ACOFS can be guided toward a particular direction by the choice of which reduced-size subset of salient features is likely to be generated. The subset size determination scheme used can be described in two ways as follows.

First, ACOFS uses a probabilistic formula modified from Muni et al. (2006) to decide the size of a subset r ($\leqslant n$) as follows:

$$P_r = \frac{n-r}{\sum_{i=1}^{l}(n-i)} \tag{2}$$

Here, P_r is maximized linearly as r is minimized, and the value of r is restricted by a constraint, namely, $2 \leqslant r \leqslant \delta$. Therefore, $r = 2,3,\ldots,\delta$, where $\delta = \mu \times n$ and $l = n - r$. Here, μ is a user-specified parameter that controls δ. Its value depends on the n for a given dataset. If δ is closed to n, then the search space of finding the salient features becomes larger, which certainly causes a high computational cost, and raises the risk that ineffective feature subsets might be generated. Since the aim of the proposed ACOFS is to select a subset of salient features within a smaller range, we prefer the length of the selected subset to be between 3 and 12 depending on the given dataset. Thus, μ is set as $\mu \in [0.1,0.6]$. We then normalize all the values of P_r in such a way that the summation of all possible values of P_r is equal to 1.

Second, ACOFS utilizes all the values of P_r for the random selection scheme mentioned in Fig. 4 to determine the size of the subset, r eventually. This selection scheme is almost similar to the classical roulette wheel procedure (Goldberg, 1989).

3.2. Subset evaluation

Subset evaluation has a significant role, along with other basic operations of ACO for selecting salient features in FS tasks. In common practices, filter or wrapper approaches are involved for evaluation tasks. However, Guyon and Elisseeff (2003) found that the performance of a wrapper approach is always better than that of a filter approach. Therefore, in the present study, we are inspired to evaluate the constructed subsets for each iteration using a feed-forward NN training scheme. Such a NN classifier is not an inherent constraint; instead of NN, any other type of classifier, such as SVM, can be used as well for this evaluation tasks. In this study, the evaluation of the subset is represented by the percentage value of NN classification accuracy (CA) for the testing set. A detailed discussion of the evaluation mechanism integrated into ACOFS as follows.

First, during training the features of a constructed subset, the NN is trained partially for τ_p epochs. Training is performed sequentially using the examples of a training set and a backpropagation (BP) learning algorithm (Rumelhart & McClelland, 1986). The number of training epochs, τ_p, is specified by the user. In partial training, which was first used in conjunction with an evolutionary algorithm (Yao & Liu, 1997), the NN is trained for a fixed number of epochs, regardless of whether the algorithm has converged on a result.

Second, check the progress of training to determine whether further training is necessary. If training error is reduced by a predefined amount, ε, after the τ_p training epochs (as mentioned in Eq. (4)), we assume that the training process has been progressing well, and that further training is thus necessary, and then proceed to the first step. Otherwise, we go to the next step for adding a hidden neuron. The error, E, is calculated as follows:

```
Random_selection
{
    generate random value h [0,1];
    sum=0;P 0=0;P1=0;
    for(r=2 to δ){
        sum=sum+Pr;
        if(h<=sum)
            break;
    }
    return r;
}
```

Fig. 4. Pseudo-code of the random selection procedure.

$$E = \frac{1}{2}\sum_{p=1}^{P}\sum_{c=1}^{C}(o_c(p) - t_c(p))^2 \quad (3)$$

where $o_c(p)$ and $t_c(p)$ are the actual and target responses of the c-th output neuron for the training example p. The symbols P and C represent the total number of examples and of output neurons in the training set, respectively. The reduction of training error can be described as follows:

$$E(t) - E(t + \tau_p) > \varepsilon, \quad t = \tau, 2\tau, 3\tau \quad (4)$$

On the other hand, in the case of adding the hidden neuron, the addition operation is guided by computing the contributions of the current hidden neurons. If the contributions are high, then we can assume that another one more hidden neuron is required. Otherwise, freeze the extension of the hidden layer size for further partial training of the NN. Computation of the contribution of previously added hidden neurons in the NN is based on the CA of the validation set. The CA can be calculated as follows:

$$CA = 100\left(\frac{P_{vc}}{P_v}\right) \quad (5)$$

where P_{vc} refers to the number of examples in the validation set correctly classified by the NN and P_v is the total number of patterns in the validation set.

At this stage, the ACOFS measures error and CA in the validation set using Eqs. (3) and (5) after every τ_p epochs of training. It then terminates training when either the validation CA decreases or the validation error increases or both are satisfied for T successive times, which are measured at the end of each of T successive τ_p epochs of training (Prechelt, 1994). Finally, we check the testing accuracy of the current NN architecture with selected hidden neurons, using the example of the testing set according to Eq. (5).

The idea behind this evaluation process is straightforward: minimize the training error, and maximize the validation accuracy. To achieve these goals, ACOFS uses a constructive approach to determine NN architectures automatically. Although other approaches, such as, pruning (Reed, 1993) and regularization (Girosi, Jones, & Poggio, 1995) could be used in ACOFS, the selection of an initial NN architecture in these approaches is difficult (Kwok & Yeung, 1997). This selection, however, is simple in the case of a constructive approach. For example, the initial network architecture in a constructive approach can consist of a hidden layer with one neuron. On the other hand, an input layer is set with r neurons, and an output layer with c neurons. More precisely, among r and c neurons, one neuron for each feature of the corresponding subset and one neuron for each class, respectively. If this minimal architecture cannot solve the given task, hidden neurons can be added one by one. Due to the simplicity of initialization, the constructive approach is used widely in multi-objective learning tasks (Lehtokangas, 2000).

3.3. Best subset selection

Generally, finding salient subsets with a reduced size is always preferable due to the low cost in hardware implementation and less time consumed in operation. Unlike other existing algorithms (e.g., Aghdam et al., 2009; Ani, 2005), in this study, the best salient feature subset is recognized eventually as a combination of the local best and global best selections as follows:

Local best selection: Determine the local best subset, $S^l(t)$ for a particular t ($t \in 1,2,3,\ldots$) iteration according to $Max(\gamma(S^k(t)))$ where $S^k(t)$ is the number of subsets constructed by k ants, and $k = 1,2,\ldots,n$.

Global best selection: Determine the global best subset (S^g), that is, the best subset of salient features from the all local best solutions in such a way that S^g is compared with the currently decided local best subset, $S^l(t)$ at every t iteration by their classification performances. If $S^l(t)$ is found better, then $S^l(t)$ is replaced by S^g. One thing is that, during this selection process, if the performances are found similar at any time, then select the one among the two, i.e., S^g and $S^l(t)$ as a best subset that has reduced size. Note that, at the first iteration $S^l(t)$ is considered as S^g.

3.4. Hybrid search process

The new hybrid search technique, incorporated in ACOFS, consists of wrapper and filter approaches. According to our best knowledge, this technique is the first adoption of the ACO-based FS approach. A significant advantage of this search technique is that ants achieve a significant ability of utilizing previous successful moves and of expressing desirability of moves towards a high-quality solution in FS. This search process is composed of two sets of newly designed rules, such as, the pheromone update rule and the heuristic information rule, which are further described as follows.

3.4.1. Pheromone update rule

Pheromone updating in the ACO algorithm is a vital aspect of FS tasks. Ants exploit features in SC that have been most suitable in prior iterations through the pheromone update rule, consisting of local update and global update. More precisely, global update applies only to those features that are a part of the best feature subset in the current iteration. It allows the features to receive a large amount of pheromone update in equal shares. The aim of global update is to encourage ants to construct subsets with a significant CA. In contrast to the global update, local update not only causes the irrelevant features to be less desirable, but also helps ants to select those features which have never been explored before. This update either decreases the strength of the pheromone trail or maintains the same level, based on whether a particular feature has been selected.

In ACOFS, a set of new pheromone update rules has been designed on the basis of two basic behaviors (that is to say, random and probabilistic) of ants during SCs. These rules have been modified from the standard rule in Aghdam et al. (2009) and Dorigo and Stutzle (2004), which aims to provide a proper balance between exploration and exploitation of ants for the next iteration. Exploration is reported to prohibit ants from converging on a common path. Actual ants also have a similar behavioral characteristic (Dorigo, Caro, & Gambardella, 1999), which is an attractive property. If different paths can be explored by different ants, then there is a higher probability that one of the ants may find a better solution, as opposed to all ants converging on the same tour.

Random case: The rule presenting in Eq. (6) is modified only in the second term, which is divided by m_i. Such a modification provides for sufficient exploration of the ants for the following constructions. The reason is that during the random behavior of the transition rule, the features are being chosen to be selected randomly in practice, instead of according to their experiences. Thus, to provide an exploration facility for the ants, the modification has been adopted as follows:

$$\tau_i(t+1) = (1-\rho)\tau_i(t) + \frac{1}{m_i}\sum_{k=1}^{n}\Delta\tau_i^k(t) + e\Delta\tau_i^g(t) \quad (6)$$

$$\Delta\tau_i^k(t) = \begin{cases} \gamma(S^k(t)) & \text{if } i \in S^k(t) \\ 0 & \text{otherwise} \end{cases}, \quad \Delta\tau_i^g(t) = \begin{cases} \gamma(S^l(t)) & \text{if } i \in S^l(t) \\ 0 & \text{otherwise} \end{cases}$$

Here, i refers to the number of feature ($i = 1,2,\ldots,n$), and m_i is the count for the specific selected feature i in the current iteration. $\Delta\tau_i^k(t)$ is the amount of pheromone received by the local update for feature i which is included in $S^k(t)$ at iteration t. Similarly, the global update, $\Delta\tau_i^g(t)$, is the amount of pheromone for feature i that is included in $S^l(t)$. Finally, ρ and e refer to the pheromone decay value, and elitist parameter, respectively.

Probabilistic case: Eq. (7) shows the modified pheromone rule for the probabilistic case. The rule is similar to the original form, but actual modification has been made only for the inner portions of the second and third terms

$$\tau_i(t+1) = (1-\rho)\tau_i(t) + \sum_{k=1}^{n} \Delta\tau_i^k(t) + e\Delta\tau_i^g(t) \tag{7}$$

$$\Delta\tau_i^k(t) = \begin{cases} \gamma(S^k(t)) \times \lambda_i & \text{if } i \in S^k(t) \\ 0 & \text{otherwise} \end{cases},$$

$$\Delta\tau_i^g(t) = \begin{cases} \gamma(S^l(t)) \times \lambda_i & \text{if } i \in S^l(t) \\ 0 & \text{otherwise} \end{cases}$$

Here, feature i is rewarded by the global update, and $\Delta\tau^g$ is in the third term, where $i \in S^l(t)$. It is important to emphasize that, i is maintained strictly here. That is, i at iteration t_t is compared with i at iteration $(t_t - \tau_p)$, where $t_t = t + \tau_p$, and $\tau_p = 1,2,3,\ldots$ In this regard, if $\gamma(S^l(t_t)) < \max(\gamma(S^l(t_t - \tau_p)),\varepsilon)$, where ε refers to the number of CAs for those local best subsets that maintain $|S^l(t_t)| = |S^l(t_t - \tau_p)|$, then a number of features, n_c are ignored to get $\Delta\tau^g$, since those features are available in $S^l(t_t)$, which causes to degrade its performance. Here, $n_c \in S^l(t_t)$ but $n_c \notin S^{lb}$, where S^{lb} provides max $(\gamma(S^l(t_t - \tau_p)),\varepsilon)$, and $|S^l(t_t)|$ implies the size of the subset $S^l(t_t)$. Note that, the aim of this restriction is to provide $\Delta\tau^g$ only to those features that are actually significant, because, global update has a vital role in selecting the salient features in ACOFS. We attempt to distinguish such salient features and to allow them to receive $\Delta\tau^g$ by imposing the above restriction.

3.4.2. Heuristic information measurement

A heuristic value, η, for each feature generally represents the attractiveness of the features, and depends on the dependency degree. It is therefore necessary to use η; otherwise, the algorithm may become too greedy, and ultimately a better solution may not be appeared (Ke et al., 2008). Here, a set of new rules is introduced for measuring heuristic information using the advantages of wrapper and filter tools. More precisely, the outcome of subset evaluations using the NN is used here as a wrapper tool, whereas the value of information gain for each feature is used as a filter tool. These rules are therefore formulated according to the random and probabilistic behaviors of the ants, which are described as follows.

Random case: In the initial iteration, while ants are involved in constructing the feature subsets randomly, the heuristic value of all features i can be estimated as follows:

$$\eta_i = \frac{1}{m_i} \sum_{k=1}^{n} \gamma(S^k(t)) \left(1 + \phi e^{-\frac{|S^k(t)|}{n}}\right) \quad \text{if } i \in S^k(t) \tag{8}$$

Probabilistic case: In the following iterations, when ants complete the feature SCs on the basis of the probabilistic behavior, the following formula is used to estimate η for all features i:

$$\eta_i = m_i \varphi_i \sum_{k=1}^{n} \gamma(S^k(t)) \lambda_i \left(1 + \phi e^{-\frac{|S^k(t)|}{n}}\right) \quad \text{if } i \in S^k(t) \tag{9}$$

In these two rules, φ_i refers to the number of a particular selected feature i that is a part of the subsets that are constructed within the currently completed iterations, except for the initial iteration. The aim of multiplying m_i and φ_i is to provide a proper exploitation capability for the ants during SCs. λ_i refers to the information gain (see the next section) for feature i. The aim of including λ is based on the following two factors: (a) reducing the greediness of some particular feature i in n during SCs, and (b) increasing the diversity between the features in n. Thus, different features may get an opportunity to be selected in the SC for different iterations, thus definitely enhancing the exploration behavior of ants. Furthermore, one additional exponential term has been multiplied by these rules in aiming for a reduced size subset. Here, ϕ is the user specified parameter that controls the exponential term. A detailed discussion on measurement of information gain is now given below.

3.4.2.1. Information gain measurement. In order to measure statistical properties of features for a given dataset, we are interested in measuring the property of information gain (IG). The aim of this measurement is to provide a statistical measure of the relevance of features to the heuristic information η of all individual features. In this regard, measuring IG is useful in the sense that the relevance of a feature can be determined statistically by the highest value of IG (Intan & Yuliana, 2009; Mitchell, 1997). A detailed description of measurement of IG can be found in Mitchell (1997). The information gain $IG(P,N_i)$ for a feature N_i ($i = 1,2,\ldots,n$) for a number of examples P of a given dataset can be defined as follows:

$$IG(P,N_i) \equiv Entropy(P) - \sum_{s \in vals(N_i)} \frac{P_s}{P} Entropy(P_s), \tag{10}$$

$$Entropy(P) = \sum_{i=1}^{c} -p_i \log_2 p_i \tag{11}$$

where $vals(N_i)$ is the set of all possible values for the feature N_i, and P_s is the subset of P for which N_i has value s, that is to say, $P_s = \{p \in P | N_i(p) = s\}$. It should be noted that P_s is the number of sub-examples for a particular value of s among the total number of examples P of the given dataset. However, in practice, measuring $Entropy(P_s)$ for the continuous random values of N_i requires a huge computational cost, which is impossible to afford in some cases. To overcome such shortcomings, continuous random variables need to be partitioned when measuring entropy. There are two main approaches for this purpose: equal distance partitioning (Battiti, 1994), and equiprobable partitioning (Fraser & Swinney, 1986). In this study, we use the equiprobable partitioning technique, described as follows.

If the distribution of the values in N_i is not known in advance, compute the mean μ and standard deviation σ. Then, cut the interval $[\mu - 2\sigma, \mu + 2\sigma]$ into q equally spaced segments, where q is the number of partitions of N_i. The values of N_i falling inside are included in the respective segments, whereas those falling outside are included in the extreme left or right segments. Each segment corresponds to a discrete value of N_i.

According to our observations, in a classification dataset, some features are more continuous, while others contain discrete values. Therefore, in this study, we attempt first to find out the discrete values for individual N_i, $vals(N_i)$, that is to say, v_i. If $v_i > 2q$, then follow the aforementioned partition technique for that particular feature only, otherwise, avoid the such partitioning process. Repeat this process for all features. Finally, measure the IG for each feature using Eqs. (10) and (11).

3.5. Computational complexity

An exact analysis of computational complexity helps in understanding the actual computational cost of an algorithm. Given that big-O notation is a prominent approach in terms of analyzing computational complexity, as shown in Kudo and Sklansky (2000), we are inspired to compute the computational cost of ACOFS. There

are seven basic steps in ACOFS, namely, information gain measurement, subset construction, subset evaluation, termination criterion, subset determination, pheromone update, and heuristic information measurement. The following paragraphs present the computational complexity of ACOFS in order to show that inclusion of different techniques does not increase computational complexity in selecting a feature subset.

(i) *Information gain measurement*: In this step, we measure information gain (IG) for each feature according to Section 3.4.2.1. If the number of total features for a given dataset is n, then the cost of measuring IG is $O(n \times P)$, where P denotes the number of examples in the given dataset. It is further mentioning that this cost is required only once, specifically, before starting the FS process.

(ii) *Subset construction*: Subset construction (SC) shows two different types of phenomena according to Eq. (1). For the random case, if the total number of features for a given dataset is n, then the cost of an ant constructing a single subset is $O(r \times n)$. Here, r refers to the size of subsets. Since the total number of ants is k, the computational cost is $O(r \times k \times n)$ operations. However, in practice, $r < n$; hence, the cost becomes $O(k \times n) \approx O(n^2)$. In terms of the probabilistic case, ACOFS uses the Eq. (1) for selecting the features in SC which shows a constant computational cost of $O(1)$ for each ant. If the number of ants is k, then the computational cost becomes $O(k)$.

(iii) *Subset evaluation*: In ACOFS, five types of operations are necessarily required for evaluating a single subset using a constructive NN training scheme: (a) partial training, (b) stopping criterion, (c) further training, (d) contribution computation, and (e) addition of a hidden neuron. The subsequent paragraphs describe these types in details.

(a) *Partial training*: We use standard BP (Rumelhart & McClelland, 1986) for training. During training each epoch BP takes $O(W)$ operations for one example. Here, W is the number of weights in the current NN. Thus, training all examples in the training set for τ_p epochs requires $O(\tau_p \times P_t \times W)$ operations, where P_t denotes the number of examples in the training set.

(b) *Stopping criterion*: During training, the stopping criterion uses either validation accuracy or validation errors for subset evaluation. Since training error is computed as a part of the training process, evaluating the termination criterion takes $O(P_v \times W)$ operations, where P_v denotes the number of examples in the validation set. Since $P_v < P_t$, $O(P_v \times W) < O(\tau_p \times P_t \times W)$.

(c) *Further training*: ACOFS uses Eq. (4) to check whether further training is necessary. The evaluation of Eq. (4) takes a constant number of computational operations $O(1)$, since the error values used in Eq. (3) have already been evaluated during training.

(d) *Contribution computation*: ACOFS computes the contribution of the added hidden neuron using Eq. (5). This computation takes $O(P_v)$ operations, which is less than $O(\tau_p \times P_t \times W)$.

(e) *Addition of a hidden neuron*: The computational cost for adding a hidden neuron is $O(r + c)$ for initializing the connection weights, where r is the number of features in the current subset, and c is the number of neurons in the output layer. Also note that $O(r + c) < O(\tau_p \times P_t \times W)$. The aforementioned computation is done for a partial training session consisting of τ_p epochs. In general, ACOFS requires a number, say M, of such partial training sessions for evaluating a single subset. Thus, the cost becomes $O(\tau_p \times M \times P_t \times W)$. Furthermore, by considering all subsets, the computational cost required is $O(k \times \tau_p \times M \times P_t \times W)$ operations.

(iv) *Termination criterion*: A termination criterion is employed in ACOFS for terminating the FS process eventually. Since only one criterion is required to be executed (i.e., the algorithm achieves a predefined accuracy, or executes a iteration threshold, I), the execution of such a criterion requires a constant computational cost of $O(1)$.

(v) *Subset determination*: ACOFS requires two steps to determine the best subset, namely, finding the local best subset, and the global best subset. In order to find the local best subset in each iteration t, ACOFS requires $O(k)$ operations. The total computational cost for finding the local best subsets thus becomes $O(k \times t)$. In order to find the global best subset, ACOFS requires $O(1)$ operations. Thus, the total computational cost for subset determination becomes $O(k \times t)$, which is less than $O(k \times \tau_p \times M \times P_t \times W)$.

(vi) *Pheromone update rule*: ACOFS executes Eqs. (6) and (7) to update the pheromone trails for each feature in terms of the random and probabilistic cases. Since the number of features is n for a given learning dataset, the computation takes $O(n)$ constant operations, which is less than $O(k \times \tau_p \times M \times P_t \times W)$.

(vii) *Heuristic information measurement*: Similar to the pheromone update operation, ACOFS uses Eqs. (8) and (9) to update the heuristic value of n features. Thereafter, the computational cost becomes $O(n)$. Note that, $O(n) \ll O(k \times \tau_p \times M \times P_t \times W)$.

In accordance with the above analysis, we can summarize the different parts of the entire computational cost as $O(n \times P) + O(n^2) + O(k) + O(k \times \tau_p \times M \times P_t \times W)$. It is important to note here that the first and second terms, namely, $n \times P$ and n^2, are the cost of operations performed only once, and are much less than $k \times \tau_p \times M \times P_t \times W$. On the other hand, $O(k) \ll O(k \times \tau_p \times M \times P_t \times W)$. Hence, the total computational cost of ACOFS is $O(k \times \tau_p \times M \times P_t \times W)$. Analyzing this, we can find that $O(\tau_p \times M \times P_t \times W)$ is similar to the cost of training a fixed network architecture using BP (Rumelhart & McClelland, 1986), and that the total cost is similar to that of other existing ACO-based FS approaches (e.g., Sivagaminathan & Ramakrishnan, 2007). Thus, we can say that incorporation of several techniques in ACOFS does not increase the computational cost.

4. Experimental studies

The performance of ACOFS has been presented in this context on eight well-known benchmark classification datasets, including the breast cancer, glass, vehicle, thyroid, ionosphere, credit card, sonar, and gene datasets; and one gene expressional classification dataset, namely, the colon cancer dataset. These datasets have been the subject of many studies in NNs and machine learning, covering examples of small, medium, high, and very high-dimensional datasets. The characteristics of these datasets, summarized in Table 1, show a considerable diversity in the number of features, classes, and examples. A detailed description of these datasets, except for colon cancer, can be obtained from the University of California Irvine Machine Learning Repository (Newman, Hettich, Blake, & Merz, 1998; Prechelt, 1994). Alon et al. (1999) describes detailed information about the colon cancer dataset. Experimental details, results, roles of subset size determination scheme in FS, the user specified parameter μ in FS, and hybrid search in FS are

Table 1
Characteristics and partitions of different classification datasets.

Dataset	Feature	Class	Example	Partition set		
				Training	Validation	Testing
Cancer	9	2	699	349	175	175
Glass	9	6	214	108	53	53
Vehicle	18	4	846	424	211	211
Thyroid	21	3	7200	3600	1800	1800
Ionosphere	34	2	351	175	88	88
Credit card	51	2	690	346	172	172
Sonar	60	2	208	104	52	52
Gene	120	3	3175	1587	794	794
Colon cancer	2000	2	62	30	16	16

described in this context. Finally, one additional experiment on ACOFS concerning performance for FS over real-world datasets mixed with some noisy features, and comparisons of ACOFS with other existing works, are also discussed in this context.

4.1. Experimental methodology

Extensive experiments have been carried out on ACOFS in order to ascertain the effectiveness of ACOFS for FS. Basically, we conducted two sets of experiments to investigate the essence of FS. In one set, we used ACOFS that selects salient feature subset during FS process. We call this set of experiment as "Average result with selected features". In the other set, FS was not performed, rather a constructive feed-forward NN training scheme was used to measure the testing CA using all features of a given dataset. We call this set of experiment as "Average result with all features". However, to accomplish the FS task suitably, three steps need to be considered, namely, dataset partitioning, reducing dimensionality of datasets, and assigning values for user-specified parameters. Regarding these issues, a detailed discussion is given in the following subsections.

4.1.1. Dataset partitioning

In this work, examples are given of nine datasets, each of which was partitioned into three subsets: a training set, a validation set, and a testing set. Distribution of the number of examples and characteristics of these datasets are reported in Table 1. Specifically, the training set was used to train and modify NN architectures; the validation set was used for terminating the training process of NNs; and the testing set was used for measuring the generalization ability of the NNs. In all datasets, the first P_t examples were used for the training set, the following P_v examples for the validation set, and the final P_s examples for the testing set. These partitions were used in accordance with benchmarking methodologies (Prechelt, 1995, 1996).

4.1.2. Dimensionality reduction of dataset

In contrast to other datasets used in this study, colon cancer is a very high-dimensional dataset containing a very large number of genes (features). Specifically, this dataset contains gene expressions of 40 tumor and 22 normal colon tissue samples, which were collected from 2000 genes out of 6500 with an affymetrix oligonucleotide array. The number of genes of colon cancer is too high to manipulate in the learning classifier, and not all genes are useful for classification (Kim & Cho, 2004). To remove such difficulties, we first reduced the dimension of the colon cancer dataset to within 100 features, using an IG measurement technique (Intan & Yuliana, 2009; Mitchell, 1997). Ordinarily, IG measurement determines statistically those features that are informative for classifying a target. On the basis of such a concept, we have used such a technique for reducing the dimension of the colon cancer dataset. Details about information gain measurement can be found in Section 3.4.2.1.

4.1.3. User-specified parameters

There are a number of user specified parameters, the values of which need to be determined for ACOFS to function suitably for the FS task. Table 2 shows parameters that are common to all datasets. Note that, these parameters are not specific to our algorithm, but are required for any ACO-based FS algorithm using a NN classifier. The parameters were chosen after a number of preliminary runs, and were not meant to be optimal. It is worth mentioning that among the parameters mentioned in Table 2, proper selection of the values of parameters α and β is helpful for achieving an effective balance between exploitation and exploration of ants in selecting salient features. For example, if $\alpha = 0$, then no pheromone information is used, that is to say, previous search experience is neglected. The search then changes to a greedy search. If $\beta = 0$, then attractiveness, the potential benefit of moves, is neglected. In this work, the values of α and β were chosen according to the suggestion of Dorigo and Stutzle (2004).

4.2. Experimental results

Table 3 shows the results of ACOFS over 20 independent runs on nine real-world benchmark classification datasets. The CA in Table 3 refers to the percentage of exact classifications produced by trained NNs on the testing set of a classification dataset. In addition, the weights of features for the above nine datasets over 20 independent runs are exhibited in Tables 4–11. On the other hand, Fig. 5 shows how the best solution was selected in ACOFS for the glass dataset. In order to observe whether the internal process of FS in ACOFS is appropriately being performed, Figs. 6–9 have been considered. Now, the following observations can be made from Tables 3–11 and Figs. 5–9.

(i) As can be seen from Table 3, ACOFS was able to select a smaller number of features for solving different datasets. For example, ACOFS selected, on average, 3.00 features from a set of 21 features in solving the thyroid dataset. It also selected, on average, 7.25 genes (features) from a set of 120 genes in solving the gene dataset. On the other hand, a very large-dimensional dataset, that of colon cancer, was preprocessed from the original one to be utilized in ACOFS. In this manner, the original 2000 features of colon cancer were reduced to within 100 features. ACOFS then obtained a small number of salient genes, 5.25 on average, from the set of 100 genes for solving the colon cancer dataset. In fact,

Table 2
Common parameters for all datasets.

Parameter	Value
Initial pheromone level for all features, τ	0.5
Initial heuristic value for all features, η	0.1
μ, used in subset size determination	0.08–0.6
Strength of pheromone level, α	1
Strength of heuristic value, β	3
Pheromone decay parameter, ρ	0.4
Exponential term control parameter, ϕ	0.1
Iteration threshold,	10–18
Accuracy threshold	Depends on dataset
Learning rate for BP algorithm	0.1–0.2
Momentum term for BP algorithm	0.5–0.9
Initial weights of NNs	−1.0 to 1.0
The number of epochs for partial training, τ	20–40
Training error threshold, λ	Depends on dataset
Training threshold for terminating NN training, T	3

Table 3
Performance of ACOFS for different classification datasets. Results were averaged over 20 independent runs. Here, n and n_s refer to the total number of original features and selected features, respectively. On the other hand, CA and SD signify the classification accuracy and standard deviation, respectively.

Dataset	Avg. result with all features				Avg. result with selected features			
	n	SD	CA (%)	SD	n_s	SD	CA (%)	SD
Cancer	9.00	0.00	97.97	0.42	3.50	1.36	98.91	0.40
Glass	9.00	0.00	76.60	2.55	3.30	1.14	82.54	1.44
Vehicle	18.00	0.00	60.71	11.76	2.90	1.37	75.90	0.64
Thyroid	21.0	0.00	98.04	0.58	3.00	1.34	99.08	0.11
Ionosphere	34.0	0.00	97.67	1.04	4.15	2.53	99.88	0.34
Credit card	51.0	0.00	85.23	0.67	5.85	1.76	87.99	0.38
Sonar	60.0	0.00	76.82	6.97	6.25	3.03	86.05	2.26
Gene	120.0	0.00	78.97	5.51	7.25	2.53	89.20	2.46
Colon cancer	100.0	0.00	59.06	5.75	5.25	2.48	84.06	3.68

Table 4
Weights of the features selected by ACOFS for the cancer and glass datasets.

Dataset	Feature								
	1	2	3	4	5	6	7	8	9
Cancer	0.186	0.042	0.129	0.142	0.129	0.2	0.115	0.042	0.015
Glass	0.258	0.045	0.258	0.107	0.06	0.015	0.182	0.06	0.015

Table 5
Weights of the features selected by ACOFS for the vehicle dataset.

Feature	1	2	4	7	9	10	11	12
Weight	0.189	0.103	0.069	0.051	0.086	0.086	0.103	0.086

Table 6
Weights of the features selected by ACOFS for the thyroid dataset.

Feature	1	7	17	19	20	21
Weight	0.052	0.052	0.332	0.1	0.069	0.15

Table 7
Weights of the features selected by ACOFS for the ionosphere dataset.

Feature	1	3	4	5	7	8	12	27	29
Weight	0.108	0.036	0.036	0.036	0.06	0.12	0.06	0.12	0.036

Table 8
Weights of the features selected by ACOFS for the credit card dataset.

Feature	5	8	29	41	42	43	44	49	51
Weight	0.042	0.06	0.034	0.051	0.17	0.111	0.128	0.034	0.12

ACOFS selected a small number of features for all other datasets having more features. Feature reduction in such datasets was several orders of magnitude (see Table 3).

(ii) The positive effect of a small number of selected features (n_s) is clearly visible when we observe the CA. For example, for the vehicle dataset, the average CA of all features was 60.71%, whereas it had been 75.90% with 2.90 features. Similarly, ACOFS produced an average CA of 86.05% with the average number of features of 6.25 substantially reduced for the sonar dataset, while the average CA had been 76.82% with all 60 features. Other similar types of scenarios can also be seen for all remaining datasets in ACOFS. Thus, it can be said that ACOFS has a powerful searching capability for providing high-quality solutions. CA improvement for such datasets was several orders of magnitude (see Table 3). Furthermore, the use of n_s caused a relatively small standard deviation (SD), as presented in Table 3 for each entry. The low SDs imply robustness of ACOFS. Robustness is represented by consistency of an algorithm under different initial conditions.

(iii) The method of determination for the final solution of a subset in ACOFS can be seen in Fig. 5. We can observe that for the performances of the local best subsets, the CAs varied

Table 9
Weights of the features selected by ACOFS for the sonar dataset.

Feature	2	9	10	11	12	15	17	18	44
Weight	0.037	0.046	0.056	0.084	0.112	0.037	0.037	0.037	0.06

Table 10
Weights of the features selected by ACOFS for the gene dataset.

Feature	22	59	60	61	62	63	64	69	70	119
Weight	0.027	0.064	0.045	0.1	0.073	0.073	0.119	0.110	0.128	0.036

Table 11
Weights of the features selected by ACOFS for the colon cancer dataset.

Feature	47	72	249	267	493	765	1247	1325	1380	1843
Weight	0.051	0.038	0.051	0.038	0.051	0.038	0.038	0.038	0.051	0.051

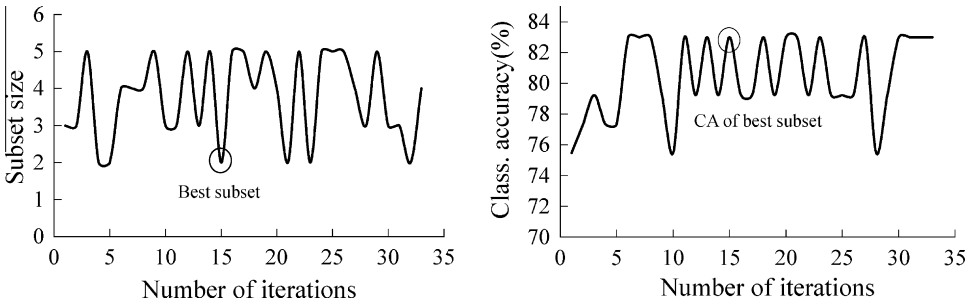

Fig. 5. Finding best subset of the glass dataset for a single run. Here, the classification accuracy is the accuracy of the local best subset.

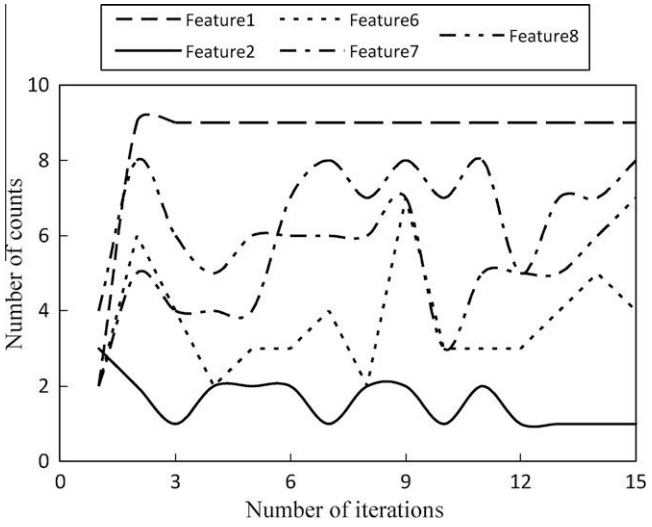

Fig. 6. Number of selections of each feature by different ants for different iterations in the glass dataset for a single run.

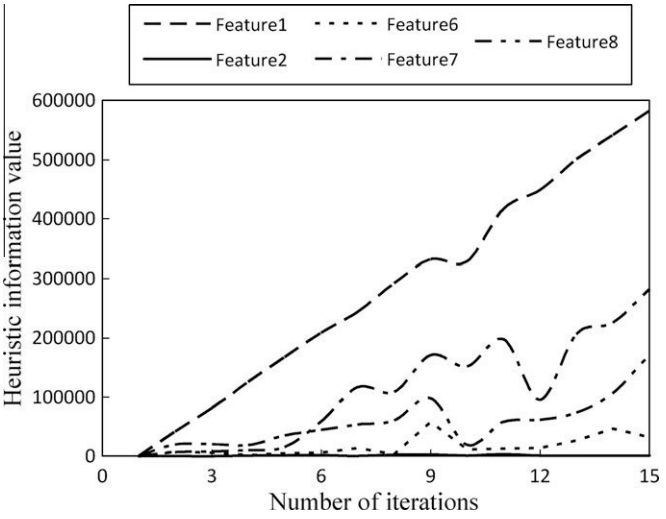

Fig. 8. Distribution of heuristic level of some selected features of the glass dataset in different iterations for a single run.

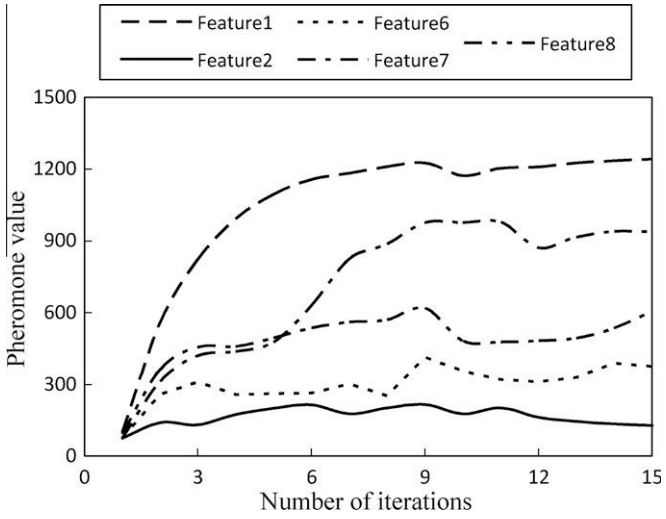

Fig. 7. Distribution of pheromone level of some selected features of the glass dataset in different iterations for a single run.

together with the size of those subsets. There were also several points where the CAs were maximized, but the best solution was selected (indicated by circle) by considering the reduced size subset. It can also be seen in Fig. 5 that CAs varied due to size variations of local best subsets in different iterations. Furthermore, different features that were included in different local best subsets caused variations in CAs.

(iv) In order to observe the manner in which how the selection of salient features in different iterations progresses in ACOFS, Fig. 6 shows the scenario of such information for the glass dataset for a single run. We can see that features 1, 7, 8, 6, and 2 received most of the selections by ants during SCs compared to the other features. The selection of features was basically performed based on the values of pheromone update (τ) and heuristic information (η) for individual features. Accordingly, those features that had higher values of τ and η ordinarily obtained a higher priority of selection, as could be seen in Figs. 7 and 8. For clarity, these figures represented five features, of which four (features 1, 7, 8, 6) had a higher rate of selection by ants during SCs and one (feature 2) had a lower rate.

(v) Upon completion of the entire FS process, the features that were most salient could be identified by means of weight computation for individual features. That is to say, features having higher weight values were more significant. On the other hand, for a particular feature to have a maximum weight value implied that the feature had the maximum number of selections by ants in any algorithm for most of the runs. Here, the weight of individual features (w_i) can be defined as follows:

$$w_i = \frac{S}{R} \tag{12}$$

where $i = 1,2,\ldots,n$, S is the number of times a particular feature is selected in all runs, and R is the total number of simulation runs. In calculating weight values, after obtaining all

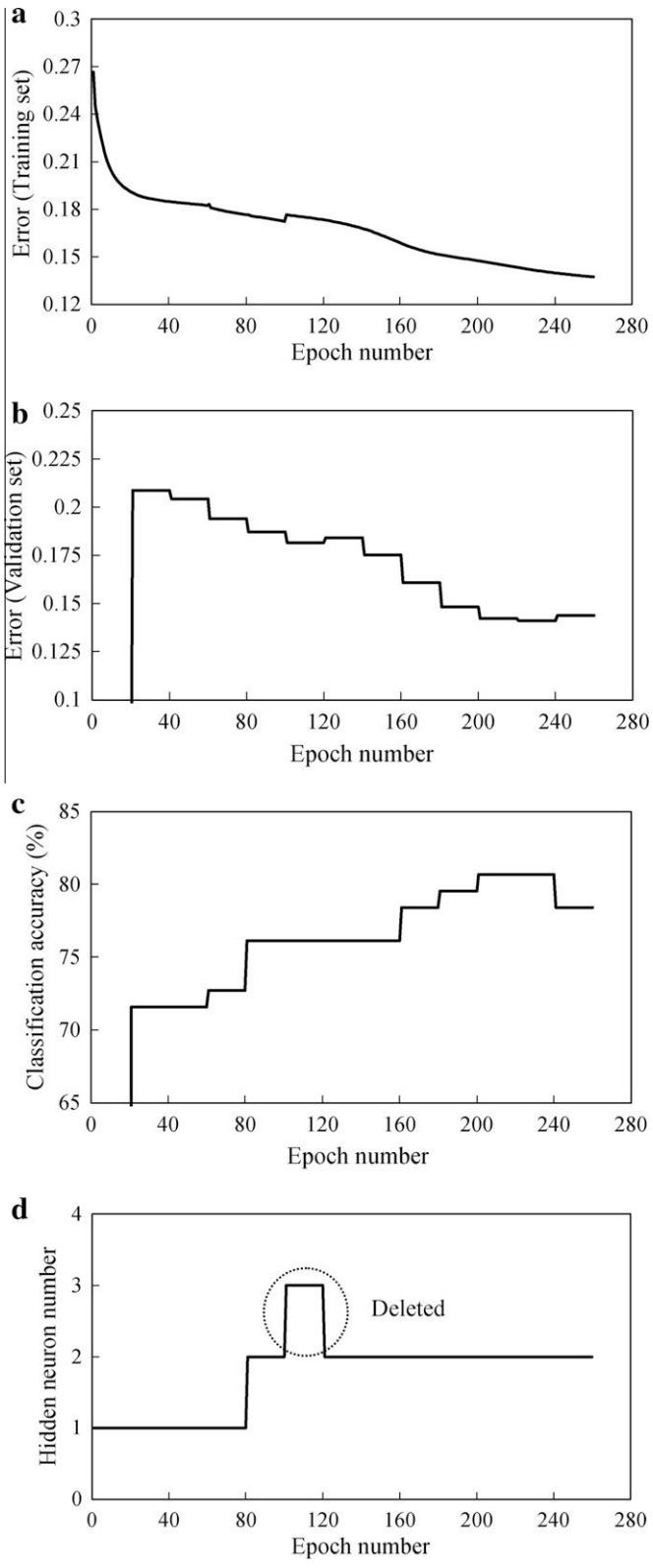

Fig. 9. Training process for evaluating the subsets constructed by ants in the ionosphere dataset: (a) training error on training set, (b) training error on validation set, (c) classification accuracy on validation set, and (d) the hidden neuron addition process.

possible values of w_i, we normalize those values such that summation of all w_i is equal to 1. Tables 4–11 show the weight of features for the cancer, glass, vehicle, thyroid, ionosphere, credit card, sonar, gene, and colon cancer datasets, respectively, over 20 independent runs. We can see in Table 4 that ACOFS selected features 6, 1, 4, 3, 5, and 7 from the cancer dataset very frequently, that these features had relatively higher weight values, and preformed well as discriminators. Similarly, our ACOFS selected features 42, 44, 51, 43, 8, and 5 as most important from the credit card dataset (Table 8), as well as features 70, 64, 69, 61, 63, 59, and 60 from the gene dataset (Table 10). Note that, weights for certain features are reported in Tables 5–11, whereas weights that were of negligible value for the rest of each dataset are not included.

(vi) Finally, we wish to note that a successful evaluation function leads to finding high-quality solutions for ACOFS in FS. Our ACOFS uses a constructive NN model that evaluates the subsets constructed by ants in each and every step during training. As training process progresses, the training error for the training set converges to a certain limit (Fig. 9(a)). However, there is an instance in which the training error increases. This is due to the addition of one unnecessary hidden neuron. Such an addition also hampers the training error on the validation set (Fig. 9(b)). Therefore, ACOFS deletes such an unnecessary hidden neuron (Fig. 9(d)) from the NN architecture, since it cannot improve the classification accuracy on the validation set (Fig. 9(c)).

4.3. Effects of subset size determination

The results presented in Table 3 show the ability of ACOFS in selecting salient features. However, the effects resulting from determining the subset size to control ants in such a manner as to construct the subset in a reduced boundary were not clear. To observe such effects, we carried out a new set of experiments. The setups of these experiments were almost exactly the same as those described before. The only difference was that ACOFS had not determined the subset size earlier using a bounded scheme; instead the size of the subset for each ant had been decided randomly.

Table 12 shows the average results of the new experiments for vehicle and credit card datasets over only 20 independent runs. The positive effects of determining the subset size during the FS process are clearly visible. For example, for the credit card dataset, the average values of n_s of ACOFS without and with subset size determination were 15.30 and 5.85, respectively. A similar scenario can also be seen for the other dataset. In terms of CAs, the average CAs for ACOFS with subset size determination were either better than or comparable to ACOFS without subset size determination for these two datasets.

4.4. Effect of μ

The essence of the proposed techniques in ACOFS can be seen in Table 3 for recognizing the subsets of salient features from the given datasets; however, the effects of the inner component μ of subset size determination (see Section 3.1) on the overall results were not clear. The reason is that the size of the subsets constructed by the ants depended roughly on the value of μ. To observe such effects, we conducted a new set of experiments. The setups of these experiments were almost exactly the same as those described before. The only difference was that the value of μ varied within a range of 0.2 to 0.94 by a small threshold value over 20 individual runs.

Tables 13 and 14 show the average results of our new experiments over 20 independent runs. The significance of the effect of varying μ can be seen from these results. For example, for the glass dataset (Table 13), the average percentage of the CA improved as

Table 12
Effect of determining subset size on the average performances of ACOFS.

Dataset	ACOFS without bounded scheme				ACOFS			
	n_s	SD	CA (%)	SD	n_s	SD	CA (%)	SD
Vehicle	6.05	4.76	75.73	0.48	2.90	1.37	75.90	0.64
Credit card	15.30	8.25	88.34	0.22	5.85	1.76	87.99	0.38

Table 13
Effect of varying the value of μ on the average performances of ACOFS for the glass dataset. The value is incremented by a threshold value of 0.01 over 20 individual runs.

Values of μ		Average performance			
Initial	Final	n_s	SD	CA (%)	SD
0.40	0.64	2.60	0.91	80.09	2.69
0.50	0.74	3.05	1.16	82.16	1.51
0.60	0.84	3.30	1.14	82.54	1.44
0.70	0.94	3.45	1.39	81.98	1.39

Table 14
Effect of varying the value of μ on the average performances of ACOFS for the ionosphere dataset. The value is incremented by a threshold value of 0.005 over 20 individual runs.

Values of μ		Average performance			
Initial	Final	n_s	SD	CA (%)	SD
0.20	0.30	4.70	2.59	99.54	0.83
0.23	0.33	3.65	2.32	99.65	0.63
0.26	0.36	4.15	2.53	99.88	0.34
0.29	0.39	6.00	3.78	99.48	0.76

the value of μ increased up to a certain point. Afterwards, the CA degraded as the value of μ increased. Thus, a subset of features was selected with a large size. A similar scenario can also be seen for the ionosphere dataset (Table 14). It is clear here that the significance of the result of FS in ACOFS depends on the value of μ. Furthermore, the determination of subset size in ACOFS is an important aspect for suitable FS.

4.5. Effect of hybrid search

The capability of ACOFS for FS can be seen in Table 3, but the effect of using hybrid search in ACOFS for FS is not clear. Therefore, a new set of experiments was carried out to observe such effects. The setups of these experiments were almost exactly as same as those described before. The only difference was that ACOFS did not use the modified rules of pheromone update and heuristic value for each feature; instead, standard rules were constructed. In such considerations, we avoided not only the incorporation of the information gain term, but also the concept of random and probabilistic behaviors, during SC for both specific rules. Furthermore, we ignored the exponential term in the heuristic measurement rule.

Table 15 shows the average results of our new experiments for the glass, credit card, sonar, and colon cancer datasets over 20 independent runs. The positive effects of using a hybrid search in ACOFS are clearly visible. For example, for the credit card dataset, the average CAs of ACOFS with and without hybrid search were 87.99% and 87.26%, respectively. A similar classification improvement for ACOFS with hybrid search was also observed for the other datasets. On the other hand, in terms of n_s, for the glass dataset, the average values of n_s of ACOFS and ACOFS without hybrid search were 3.30 and 4.05, respectively. For the other datasets it was also found that ACOFS selected a smaller number of salient features. We used *t*-test here to determine whether the difference of classification performances between ACOFS and ACOFS without hybrid search was statistically significant. We found that ACOFS performed significantly better than ACOFS without local search operation at a 95% confidence level for all the datasets except for the colon cancer dataset. On the other hand, the *t*-test was also used here to determine whether the difference in performances between the above two approaches with regard to selecting a reduced number of salient features was statistically significant. We found that ACOFS was significantly better than ACOFS without hybrid search at a 95% confidence level for all four datasets.

In order to understand precisely how hybrid search plays an important role in ACOFS for FS tasks, a set of experiments was additionally conducted. The setups of these experiments were similar to those described before, and different initial conditions were maintained constant between these two experiments. Figs. 10 and 11 show the CAs of ACOFS without and with hybrid search, respectively. These CAs were produced by local best subsets in different iterations of a single run. The positive role of using hybrid local search in ACOFS can clearly be seen in these figures. In Fig. 10, we can see that a better CA was found only in the initial iteration because of the rigorous survey by the ants in finding salient features. For the next iterations, the CAs fluctuated up to a higher iteration, 19, but were not able to reach a best state. This occurred due to the absence of hybrid search, which resulted in a weak search in ACOFS. The opposite scenario can be seen in Fig. 11, where the search was sufficiently powerful that by a very low number of iterations, 5, ACOFS was able to achieve the best accuracy (99.42%) of the salient feature subset. Thereafter, ACOFS terminated the searching of salient features. The reason for such a high performance of FS was just the incorporation of the hybrid search.

4.6. Performance on noisy features

The results presented in Table 3 exhibit the ability of ACOFS to select salient features from real-valued datasets. In this study, we examine the sensitivity of ACOFS to noisy features that have been synthetically inserted into a number of real-valued datasets. In order to generate these noisy features, we followed the process dis-

Table 15
Effect of considering hybrid search on average performances of ACOFS. Results were averaged over 20 independent runs.

Dataset	ACOFS without hybrid search				ACOFS			
	n_s	SD	CA (%)	SD	n_s	SD	CA (%)	SD
Glass	4.05	1.35	81.22	1.39	3.30	1.14	82.54	1.44
Credit card	6.15	2.21	87.26	0.66	5.85	1.76	87.99	0.38
Sonar	6.50	2.80	84.42	3.03	6.25	3.03	86.05	2.26
Colon cancer	6.35	4.05	82.18	4.08	5.25	2.48	84.06	3.68

Fig. 10. Classification accuracies (CAs) of the cancer dataset without considering hybrid search for a single run. Here, a CA is the accuracy of a local best subset.

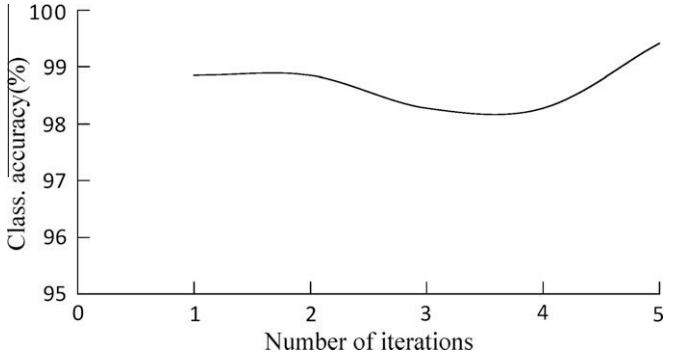

Fig. 11. Classification accuracies (CAs) of the cancer dataset in ACOFS for a single run. Here a CA is the accuracy of a local best subset.

cussed in Muni et al. (2006). Briefly, at first, we considered four features, namely, $f_{n1}, f_{n2}, f_{n3}, f_{n4}$ and the values of these respective features were generated randomly. Specifically, the values of f_{n1} and f_{n2} were bound up to [0,1] and [−1,+1], respectively. For the domains of f_{n3} and f_{n4}, we first randomly selected two different features from the datasets. Subsequently, the data points of these two selected features were taken as a random basis for use in the domains of $f_{n3}$and f_{n4}.

Table 16 shows the average performances of ACOFS on the real-valued datasets of cancer and glass mixed with noisy features over 20 independent runs. The ability of ACOFS for FS over real-valued datasets can also be found in Table 3. In comparing Tables 3 and 16, the following observations can be made. For the glass dataset, the average CAs with and without noisy features were 81.69% and 82.54%, respectively. On the other hand, in terms of n_s, the average values were 4.45 and 3.30, respectively. A similar scenario can also be found for the cancer dataset. Thus, it is clear that ACOFS has a strong ability to select the salient features from real-valued datasets even with a mixture of noisy features. We can observe that ACOFS selected a slightly higher average number of salient features from the glass dataset with noisy features. The reason is that, adding the noisy features created confusion in the feature space. This may assist our ACOFS in selecting a greater number of noiseless features to resolve the confusion in the feature space caused by the noisy features.

Table 16
Performances of ACOFS for noisy datasets. Results were averaged over 20 independent runs.

Dataset	With all features				With selected features			
	n_s	SD	CA (%)	SD	n_s	SD	CA (%)	SD
Cancer	13.00	0.00	97.80	0.89	3.80	1.80	98.74	0.46
Glass	13.00	0.00	73.86	2.81	4.45	1.71	81.69	2.31

4.7. Comparisons

The results of ACOFS obtained on nine real-world benchmark classification datasets are compared here with the results of various existing FS algorithms (i.e., ACO-based and non ACO-based) as well as with a normal ACO-based FS algorithm, as reported in Tables 17–19. The various FS algorithms are as follows: ACO-based hybrid FS ($ACOFS_S$; Sivagaminathan & Ramakrishnan, 2007), ACO-based attribute reduction (ACOAR; Ke et al., 2008), genetic programming for FS (GPFS; Muni et al., 2006), hybrid genetic algorithm for FS (HGAFS; Huang et al., 2007), MLP-based FS method (MLPFS; Gasca et al., 2006), constructive approach for feature selection (CAFS; Kabir et al., 2010), and artificial neural net input gain measurement approximation (ANNIGMA; Hsu et al., 2002). The results reported in these tables are over 20 independent runs. In comparing these algorithms, we have mainly used two parameters: classification accuracy (CA) and the number of selected features (n_s).

4.7.1. Comparison with other works

The comparisons between eight FS algorithms represent a wide range of FS techniques. Five of the FS techniques, namely, ACOFS, $ACOFS_S$, ACOAR, GPFS, and HGAFS, use global search strategies for FS. Among them, ACOFS, $ACOFS_S$, and ACOAR use the ant colony optimization algorithm. HGAFS uses a GA in finding salient features, and GPFS uses genetic programming, a variant of GA. For the remaining three FS techniques, namely, MLPFS, ANNIGMA and CAFS; MLPFS and ANNIGMA use backward selection strategy for finding salient features, while CAFS uses forward selection strategy. For evaluating the feature subset, ACOFS, $ACOFS_S$, MLPFS, CAFS, and ANNIGMA use a NN for classifiers, while GPFS and HGAFS use a decision tree and support vector machine, respectively, for classifiers, and ACOAR uses rough set theory by calculating a dependency degree. ACOFS, and CAFS uses a training set, validation set and testing set, while $ACOFS_S$ and ANNIGMA use only a training set and testing set. MLPFS and GPFS use 10-fold cross-validation. A similar method, that is, k-fold cross-validation, is used in HGAFS, where k refers to a value ranging from 2 to 10, depending on the given dataset scale. The aforementioned algorithms not only use different data partitions, but also employ a different number of independent runs in measuring average performances. For example, ANNIGMA and CAFS use 30 runs, ACOFS uses 20 runs, and MLPFS and GPFS use 10 runs. It is important to note that no further information regarding the number of runs has been mentioned in the literature for $ACOFS_S$ and HGAFS.

Table 17
Comparisons between ACOFS, $ACOFS_S$ (Sivagaminathan & Ramakrishnan, 2007), ACOAR (Ke et al., 2008). Here, "–" means not available.

Dataset		Comparison		
		ACOFS	$ACOFS_S$	ACOAR
Cancer	n_s	3.50	12.00	
	CA (%)	98.91	95.57	
Thyroid	n_s	3.00	14.00	–
	CA (%)	99.08	94.50	–
Credit card	n_s	5.85	–	8.00
	CA (%)	87.99	–	–
Colon cancer	n_s	5.25	–	8.00
	CA (%)	84.06	–	59.5

Table 18
Comparisons between ACOFS, GPFS (Muni et al., 2006), HGAFS (Huang et al., 2007), MLPFS (Gasca et al., 2006), CAFS (Kabir et al., 2010), and ANNIGMA (Hsu et al., 2002). Here, "–" means not available.

Dataset		Comparison					
		ACOFS	GPFS	HGAFS	MLPFS	CAFS	ANNIGMA
Cancer	n_s	3.50	2.23	3.00	8.00	6.33	5.80
	CA (%)	98.91	96.84	94.24	89.40	98.76	96.50
Glass	n_s	3.30	–	5.00	8.00	4.73	–
	CA (%)	82.54	–	65.51	44.10	76.91	–
Vehicle	n_s	2.90	5.37	11.00	13.00	2.70	–
	CA (%)	75.90	78.45	76.36	74.60	74.56	–
Ionosphere	n_s	4.15	–	6.00	32	6.73	9.00
	CA (%)	99.88	–	92.76	90.60	96.55	90.20
Credit card	n_s	5.85	–	1.00	–	–	6.70
	CA (%)	87.99	–	86.43	–	–	88.00
Sonar	n_s	6.25	9.45	15.00	29.00	–	–
	CA (%)	86.05	86.26	87.02	59.10	–	–
Colon cancer	n_s	5.25	–	6.00	–	–	–
	CA (%)	84.06	–	86.77	–	–	–

Table 19
Comparisons between ACOFS and NACOFS. Here, NACOFS refers to the normal ACO-based FS algorithm.

Dataset	Comparison							
	ACOFS				NACOFS			
	n_s	SD	CA	SD	n_s	SD	CA	SD
Cancer	3.50	1.36	98.91	0.40	4.50	0.97	98.77	0.37
Glass	3.30	1.14	82.54	1.44	4.60	1.01	80.66	1.44
Ionosphere	4.15	2.53	99.88	0.34	11.45	6.17	99.88	0.34
Credit card	5.85	1.76	87.99	0.38	22.85	6.01	88.19	0.45

We can see in Table 17 that ACOFS produced the best solutions in terms of a reduced number of selected features, and the best CA in comparison with the two ACO-based FS algorithms, namely, $ACOFS_S$ and ACOAR, for all four datasets. Furthermore, the results produced by ACOFS shown in Table 18 represented the best CA among the other algorithms for all four datasets. For the remaining three datasets, while HGAFS achieved the best CA for two datasets, GPFS achieved the best CA for one dataset. Note that, ACOFS and ANNIGMA jointly achieved the best CA for the credit card dataset. In terms of n_s, ACOFS selected the smallest number of features for four out of seven datasets, and the second smallest for two dataset; that is to say, CAFS and HGAFS. In a close observation, ACOFS achieved the smallest n_s, which resulted in the best CAs for the glass and ionosphere datasets in comparison with the other five algorithms (see Table 18).

Significantly, it can be said that FS improves the performance of classifiers by ignoring irrelevant features in the original feature set. An important task in such a process is to capture necessary information in selecting salient features; otherwise, the performance of classifiers might be degraded. For example, for the cancer dataset, GPFS selected the smallest feature subset consisting of 2.23 features, but achieved a lower CA. On the other hand, ACOFS selected a slightly larger feature subset that provided a better CA compared to others for the cancer dataset. In fact, the results presented for other algorithms in Table 18 indicate that having the smallest or largest feature subset did not guarantee performing with the best or worst CA.

4.7.2. Comparison with normal ACO based FS algorithm

In this context, we use a normal ACO algorithm for solving FS, considering similar steps as incorporated in ACOFS, except for a number of differences. We call this algorithm "NACOFS". In NACOFS, issues of guiding the ants and forcing the ants during SC were not considered. Instead, the ants followed a process for SC where the size of subsets was fixed for each iteration and increased at a fixed rate for following iterations. On the other hand, hybrid search was not used here; that is to say, the concept of random and probabilistic behavior was not considered, including the incorporation of information gain in designing the pheromone update rule and heuristic information measurement rule.

We can see in Table 19 that the results produced by ACOFS achieved the best CA compared to NACOFS for three out of four datasets. For the remaining dataset, NACOFS achieved the best result. In terms of n_s, ACOFS selected the smallest number of features for the all four datasets, while NACOFS selected subsets of bulky size. Between these two algorithms, the performances of the CAs seemed to be similar, but the results of the numbers of selected features were very different. The performance of ACOFS was also found to be very consistent, exhibiting a low standard deviation (SD) under different experimental setups.

4.8. Discussion

This section briefly explains the reason that the performance of ACOFS was better than those of the other ACO-based FS algorithms compared in Table 17. There are three major differences that might contribute to the better performance of ACOFS compared to the other algorithms.

The first reason is that ACOFS uses a bounded scheme to determine the subset size, while $ACOFS_S$, ACOAR, and other ACO-based FS algorithms (e.g., Aghdam et al., 2009; Ani, 2005; Kanan et al., 2007; Khushaba et al., 2008; Robbins et al., 2008) do not use such a scheme. It is now clear that without a bounded scheme, ants are free to construct subsets of bulky size. Accordingly, there is a high possibility of including a number of irrelevant features in the constructed subsets. Using the bounded scheme with assistance from other techniques, ACOFS includes the most highly salient features in a reduced number, although it functioned upon a wide range of feature spaces. As shown in Table 17, ACOFS selected, on average, 3.00 salient features, while $ACOFS_S$selected 14.00 features, on average, from the thyroid dataset. For the remaining other three datasets, ACOFS also selected a very small number of salient features. The benefit of using the bounded scheme can also be seen from the results of the selected subsets in ACOFS.

The second reason is the new hybrid search technique integrated in ACOFS. The algorithms $ACOFS_S$, ACOAR and others do

not use such a hybrid search technique in performing pheromone update and heuristic information measurement. The benefit of adopting the hybrid search in ACOFS can clearly be seen in Figs. 10 and 11. These figures show that ACOFS achieved a powerful and faster searching capability in finding salient features in the feature space. The above advantage can also be seen in Tables 17 and 18. We found that ACOFS had a remarkable capability to produce significant classification performances from different datasets using a reduced number of salient features.

The third reason is that ACOFS used a constructive approach for determining appropriate architectures, that is to say, an appropriate size of the hidden layer for the NN classifiers. The NN then evaluated the subsets constructed by the ants in each iteration during training. The existing ACO-based FS approaches (for example, Sivagaminathan & Ramakrishnan, 2007) often ignored the above issue of the NN classifiers. Furthermore, a number of other approaches (for example, Aghdam et al., 2009; Ani, 2005) often ignored the classifier portions to consider any heuristic methodology by which the activity of the classifiers could be improved for evaluating the subsets effectively. Furthermore, most ACO-based FS approaches performed the pheromone update rule based on classifier performances in evaluating the subsets. In this sense, the evaluation function was one of the most crucial portions in these approaches for FS. However, the most common practice was to choose the number of hidden neurons in the NN randomly. Thus, the random selection of hidden neurons affected the generalization performances of the NNs. Furthermore, the entire FS process was eventually affected, resulting in ineffective solutions in FS. It is also important to say that the performance of any NN was greatly dependent on the architecture (Reed, 1993; Yao & Liu, 1997). Thus, automatic determination of the number of hidden neurons' lead to providing a better solution for FS in ACOFS.

5. Conclusions

In this paper, an efficient hybrid ACO-based FS algorithm has been presented. Since ants are the foremost strength of an ACO algorithm, guiding the ants in the correct directions is a critical requirement for high-quality solutions. Accordingly, ACOFS guides ants during SC by determining the subset size. Furthermore, new sets of pheromone update and heuristic information measurement rules for individual features bring out the potential of the global search capability of ACOFS.

Extensive experiments have been carried out in this paper to evaluate how well ACOFS has performed in finding salient features on different datasets (see Table 3). It is observed that a set of high-quality solutions for FS was found from small, medium, large, and very large dimensional datasets. The results of the low standard deviations of the average classification accuracies, as well as the average number of selected features, showed the robustness of this algorithm. On the other hand, in comparison with seven prominent FS algorithms (see Tables 17 and 18), with only a few exceptions, ACOFS outperformed the others in terms of a reduced number of selected features and best classification performances. Furthermore, the estimated computational complexity of this algorithm reflected that incorporation of several techniques did not increase the computational cost during FS in comparison with other ACO-based FS algorithms (see Section 3.5).

We can see that there are a number of areas where ACOFS failed to improve performances in terms of number of selected features and classification accuracies. Accordingly, more suitable heuristic schemes are necessary in order to guide the ants appropriately. In the current implementation, ACOFS has a number of some user-specified parameters, given in Table 2, which are common in the field of ACO algorithm using NNs for FS. Further tuning of the user-specified parameters related to ACO provides some scope for further investigations in the future. On the other hand, among these parameters, μ, used in determining the subset size, was sensitive to moderate change, according to our observations. One of the future improvements to ACOFS could be to reduce the number of parameters, or render them adaptive.

Acknowledgements

This research project was supported by grants to K.M. from the Japanese Society for Promotion of Sciences, and the University of Fukui.

References

Abe, S. (2005). Modified backward feature selection by cross validation. In *Proceedings of the European symposium on artificial neural networks* (pp. 163–168).

Aghdam, M. H., Aghaee, N. G., & Basiri, M. E. (2009). Text feature selection using ant colony optimization. *Expert Systems with Applications, 36*, 6843–6853.

Alon, U., Barkai, N., Notterman, D. A, Gish, K., Ybarra, S., Mack, D., et al. (1999). Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. In *Proceedings of the national academy of sciences, USA* (Vol. 96, pp. 6745–6750).

Ani, A. (2005). Feature subset selection using ant colony optimization. *International Journal of Computational Intelligence, 2*, 53–58.

Back, A. D., & Trappenberg, T. P. (2001). Selecting inputs for modeling using normalized higher order statistics and independent component analysis. *IEEE Transactions on Neural Network, 12*(3), 612–617.

Battiti, R. (1994). Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on Neural Networks, 5*(4), 537–550.

Caruana, R., & Freitag, D. (1994). Greedy attribute selection. In *Proceedings of the 11th international conference of machine learning*. USA: Morgan Kaufman.

Caruana, R., & Sa, V. D. (2003). Benefitting from the variables that variable selection discards. *Journal of Machine Learning Research, 3*, 1245–1264.

Chakraborty, D., & Pal, N. R. (2004). A neuro-fuzzy scheme for simultaneous feature selection and fuzzy rule-based classification. *IEEE Transactions on Neural Networks, 15*(1), 110–123.

Chow, T. W. S., & Huang, D. (2005). Estimating optimal feature subsets using efficient estimation of high-dimensional mutual information. *IEEE Transactions on Neural Network, 16*(1), 213–224.

Dash, M., & Liu, H. (1997). Feature selection for classification. *Intelligent Data Analysis* (1), 131–156.

Dorigo, M., Caro, G. D., & Gambardella, L. M. (1999). Ant algorithm for discrete optimization. *Artificial life, 5*(2), 137–172.

Dorigo, M., & Stutzle, T. (2004). *Ant colony optimization*. Cambridge, MA: MIT Press.

Filippone, M., Masulli, F., & Rovetta, S. (2006). Supervised classification and gene selection using simulated annealing. *International Joint Conference on Neural Networks*, 3566–3571.

Fraser, A. M., & Swinney, H. L. (1986). Independent coordinates for strange attractors from mutual information. *Physical Review A, 33*(2), 1134–1140.

Gasca, E., Sanchez, J. S., & Alonso, R. (2006). Eliminating redundancy and irrelevance using a new MLP-based feature selection method. *Pattern Recognition, 39*, 313–315.

Girosi, F., Jones, M., & Poggio, T. (1995). Regularization theory and neural networks architectures. *Neural Computation, 7*(2), 219–269.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley Press.

Guan, S., Liu, J., & Qi, Y. (2004). An incremental approach to contribution-based feature selection. *Journal of Intelligence Systems, 13*(1).

Guyon, I., & Elisseeff, A. (2003). An Introduction to variable and feature selection. *Journal of Machine Learning Research, 3*, 1157–1182.

Hall, M. A. (2000). Correlation-based feature selection for discrete and numeric class machine learning. In *Seventeenth international conference on machine learning*.

Hsu, C., Huang, H., & Schuschel, D. (2002). The ANNIGMA-wrapper approach to fast feature selection for neural nets. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics, 32*(2), 207–212.

Huang, J., Cai, Y., & Xu, X. (2007). A hybrid genetic algorithm for feature selection wrapper based on mutual information. *Pattern Recognition Letters, 28*, 1825–1844.

Intan, R., & Yuliana, O. Y. (2009). Fuzzy decision tree induction approach for mining fuzzy association rules. In *Sixteenth international conference on neural information processing* (pp. 720–728).

Jensen, R. (2005). *Combining rough and fuzzy sets for feature selection*. Ph.D. dissertation, School of Information, Edinburgh University.

Kabir, M. M, Islam, M. M., & Murase, K. (2008). A new wrapper feature selection approach using neural network. In *Proceedings of joint 4th international conference on soft computing and intelligent systems and 9th international symposium on advanced intelligent systems, Japan* (pp. 1953–1958).

Kabir, M. M., Islam, M. M., & Murase, K. (2010). A new wrapper feature selection approach using neural network. *Neurocomputing, 73*, 3273–3283.

Kabir, M. M, Shahjahan, M., & Murase, K. (2009). An efficient feature selection using ant colony optimization algorithm. In *Sixteenth international conference on neural information processing* (pp. 242–252).

Kambhatla, N., & Leen, T. K. (1997). Dimension reduction by local principal component analysis. *Neural Computation, 9*(7), 1493–1516.

Kanan, H. R., Faez, K., & Taheri, S. M. (2007). Feature selection using ant colony optimization (ACO): A new method and comparative study in the application of face recognition system. In *International conference on data mining* (pp. 63–76).

Ke, L., Feng, Z., & Ren, Z. (2008). An efficient ant colony optimization approach to attribute reduction in rough set theory. *Pattern Recognition Letters, 29*, 1351–1357.

Khushaba, R. N., Alsukker, A., Ani, A. A., & Jumaily, A. A. (2008). Enhanced feature selection algorithm using ant colony optimization and fuzzy memberships. In *Proceedings of the sixth IASTED international conference on biomedical engineering* (pp. 34–39).

Kim, K., & Cho, S. (2004). Prediction of colon cancer using an evolutionary neural network. *Neurocomputing, 61*, 361–379.

Kudo, M., & Sklansky, J. (2000). Comparison of algorithms that select features for pattern classifiers. *Pattern Recognition, 33*, 25–41.

Kwok, T. Y., & Yeung, D. Y. (1997). Constructive algorithms for structure learning in feed-forward neural networks for regression problems. *IEEE Transactions on Neural Networks, 8*, 630–645.

Lai, C., Reinders, M. J. T., & Wessels, L. (2006). Random subspace method for multivariate feature selection. *Pattern Recognition Letters, 27*, 1067–1076.

Lehtokangas, M. (2000). Modified cascade-correlation learning for classification. *IEEE Transactions on Neural Networks, 11*, 795–798.

Liu, H., & Tu, L. (2004). Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering, 17*(4), 491–502.

Mao, K. Z. (2002). Fast orthogonal forward selection algorithm for feature subset selection. *IEEE Transactions on Neural Network, 13*(5), 1218–1224.

Mitchell, T. M. (1997). *Machine learning*. McGraw-Hill.

Muni, D. P., Pal, N. R., & Das, J. (2006). Genetic programming for simultaneous feature selection and classifier design. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics, 36*(1), 106–117.

Newman, D. J., Hettich, S., Blake, C. L., & Merz, C. J. (1998). *UCI repository of machine learning databases*. Dept. of Information and Computer Sciences, University of California, Irvine. <http://www.ics.uci.edu/~mlearn/MLRepository.html>.

Oh, I., Lee, J., & Moon, B. (2004). Hybrid genetic algorithms for feature selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 26*(11), 1424–1437.

Pal, N. R., & Chintalapudi, K. (1997). A connectionist system for feature selection. *International Journal of Neural, Parallel and Scientific Computation, 5*, 359–381.

Peng, H., Long, F., & Ding, C. (2003). Overfitting in making comparisons between variable selection methods. *Journal of Machine Learning Research, 3*, 1371–1382.

Prechelt, L. (1994). *PROBEN1 – A set of neural network benchmark problems and benchmarking rules*. Technical report 21/94, Faculty of Informatics, University of Karlsruhe.

Prechelt, L. (1995). Some notes on neural learning algorithm benchmarking. *Neurocomputing, 9*(3), 343–347.

Prechelt, L. (1996). A quantitative study of experimental evaluations of neural network learning algorithms. *Neural Network, 9*(3), 457–462.

Pudil, P., Novovicova, J., & Kittler, J. (1994). Floating search methods in feature selection. *Pattern Recognition Letters, 15*(11), 1119–1125.

Rakotomamonjy, A. (2003). Variable selection using SVM-based criteria. *Journal of Machine Learning Research, 3*, 1357–1370.

Reed, R. (1993). Pruning algorithms — A survey. *IEEE Transactions on Neural Networks, 4*(5), 740–747.

Robbins, K. R., Zhang, W., & Bertrand, J. K. (2008). The ant colony algorithm for feature selection in high-dimension gene expression data for disease classification. *Journal of Mathematical Medicine and Biology*, 1–14.

Rumelhart, D. E., & McClelland, J. (1986). *Parallel distributed processing*. MIT Press.

Setiono, R., & Liu, H. (1997). Neural network feature selector. *IEEE Transactions on Neural Networks, 8*, 654–662.

Sindhwani, V., Rakshit, S., Deodhare, D., Erdogmus, D., Principe, J., & Niyogi, P. (2004). Feature selection in MLPS and SVMs based on maximum output information. *IEEE Transactions on Neural Network, 15*(4), 937–948.

Sivagaminathan, R. K., & Ramakrishnan, S. (2007). A hybrid approach for feature subset selection using neural networks and ant colony optimization. *Expert Systems with Applications, 33*, 49–60.

Straceezzi, D. J., & Utgoff, P. E. (2004). Randomized variable elimination. *Journal of Machine Learning Research, 5*, 1331–1362.

Verikas, A., & Bacauskiene, M. (2002). Feature selection with neural networks. *Pattern Recognition Letters, 23*, 1323–1335.

Wang, X., Yang, J., Teng, X., Xia, W., & Jensen, R. (2007). Feature selection based on rough sets and particle swarm optimization. *Pattern Recognition Letters, 28*, 459–471.

Wang, L., Zhou, N., & Chu, F. (2008). A general wrapper approach to selection of class-dependent features. *IEEE Transactions on Neural Networks, 19*(7), 1267–1278.

Yang, J. H., & Honavar, V. (1998). Feature subset selection using a genetic algorithm. *IEEE Intelligent Systems, 13*(2), 44–49.

Yao, X., & Liu, Y. (1997). A new evolutionary system for evolving artificial neural networks. *IEEE Transactions on Neural Networks, 8*(3), 694–713.