Information Sciences 565 (2021) 105–127

Contents lists available at ScienceDirect

Information Sciences

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

TAGA: Tabu Asexual Genetic Algorithm embedded in a filter/filter feature selection approach for high-dimensional data

Sadegh Salesi [a], Georgina Cosma [b,*], Michalis Mavrovouniotis [c]

[a] Department of Computing, School of Science and Technology, Nottingham Trent University, Nottingham, UK
[b] Department of Computer Science, School of Science, Loughborough University, UK
[c] KIOS Research and Innovation Center of Excellence, Department of Electrical and Computer Engineering, University of Cyprus, Nicosia, Cyprus

ARTICLE INFO

Article history:
Received 15 September 2019
Received in revised form 15 October 2020
Accepted 11 January 2021
Available online 26 January 2021

ABSTRACT

Feature selection is the process of selecting an optimal subset of features required for maintaining or improving the performance of data mining models. Recently, hybrid filter/wrapper feature selection methods have shown promising results for high-dimensional data. However, filter/wrapper methods lack of generalisation power, which enables the selected features to be trainable over different classifiers without having to repeat the feature selection process. To address the generalisation power problem, this paper proposes a novel evolutionary-based filter feature selection algorithm that is sequentially hybridised with the Fisher score filter algorithm in a new hybrid framework called filter/filter. The proposed algorithm is based on a long-term memory Tabu Search combined with an Asexual (i.e. mutation-based) Genetic Algorithm (TAGA). TAGA benefits from a new integer-encoded solution representation, a novel mutation operator, a new tabu list encoding scheme, and uses a minimum redundancy maximum relevance information theory-based criterion as the fitness function. Experiments were carried out on various high-dimensional datasets including image, text, and biological data. The goodness of the selected subsets was evaluated using different classifiers and the experimental results demonstrate that TAGA outperforms other conventional and state-of-the-art feature selection algorithms.

1. Introduction

Feature selection reduces the dataset, and this also reduces the complexity of machine learning models, in terms of the time and computational processing power, that they require to learn the data [1]. Traditional feature selection approaches are commonly classified into: filter, wrapper and embedded [2].

In filter approaches, a statistical measure is applied to assign a score to each feature according to its correlation with other features and the target variable, and thereafter the features with the highest scores are selected [3]. Because filter approaches are classifier-independent they have high generalisation power over a wide range of classifiers, enabling the selected features to be trainable over different classifiers – and hence eliminating the need to repeat the feature selection process when replacing the classifier. Furthermore, filter approaches have low complexity and they are easily applicable to

* Corresponding author.
E-mail address: g.cosma@lboro.ac.uk (G. Cosma).

https://doi.org/10.1016/j.ins.2021.01.020
0020-0255/

high-dimensional datasets. One important limitation of filter approaches is that in highly corrected datasets a redundant subset of features may be selected and this can result in a subset of features that is unsuitable for training a classifier [1].

In wrapper approaches, the classification ability of the machine learning algorithm is used to evaluate the subset of features which are then mutually compared to select the best subset. In contrast to other feature selection approaches, the ability of wrapper approaches is often higher in terms of classification accuracy, but wrapper approaches require high computational effort when they are applied to high-dimensional datasets [4]. Moreover, the generalisation power of wrapper-based approaches over different classifiers is relatively weak because the selected subset of features is biased towards the classifier used in the wrapper approach [5].

Embedded approaches take advantage of both filter and wrapper approaches [6], and aim to select highly discriminant features. Embedded approaches remove noisy features by analysing and measuring the effects of the selected features upon constructing a classifier. Embedded approaches learn which features best contribute to the classification accuracy of the model while the model is being created. Although the complexity of embedded approaches is lower than that of the wrapper approaches, embedded approaches are not suitable for high-dimensional datasets since they inherit the disadvantages of wrapper approaches.

Recent approaches include hybrid feature selection approaches that combine the best properties of filter and wrapper approaches [7]. Hybrid approaches are known to be more suitable for high-dimensional datasets compared to the other types of feature selection approaches [2,8]. A typical hybrid feature selection approach, also called filter/wrapper, consists of two stages. In the first stage, a filter approach is used to find the most discriminating features in order to reduce the dimensionality of the feature space. In the second stage, a wrapper approach is employed to find the best subset using the features identified in the first stage. Consequently, hybrid approaches inherit the poor performance in terms of generalisation power (i.e. choosing a different classifier will return a different subset of features) of wrapper approaches which increases the complexity of finding the optimal subset of features. Therefore, when developing a feature selection algorithm, it is important to develop the one for which the selected features can provide acceptable performance over a wide range of classifiers. This is of significant importance particularly in tasks when the best performing classifier for the dataset is not known in advance.

One potential solution to the generalisation power problem of filter/wrapper approaches is to combine two filter approaches in a sequential framework which is called a filter/filter approach. To the best of the authors' knowledge, this is the first paper employing this type of hybrid approach. The most important difference between a filter/wrapper approach and a filter/filter approach is that in the second stage of the latter approach a statistical metric (e.g. correlation and information theory-based) is used as the fitness function by the feature selection algorithm to evaluate the goodness of subsets (rather than a classifier which is used in filter/wrapper). Similar to filter/wrapper approaches, a filter/filter approach consists of two stages in which a score-based filter algorithm is sequentially hybridised with an evolutionary-based filter algorithm.

In this paper, a novel evolutionary-based feature selection algorithm is proposed which is embedded into a two-stage filter/filter approach. In the first stage, the Fisher score feature selection algorithm [9–11] which is computationally cost effective, is applied to reduce the size of the datasets by filtering out the most promising features which are then fed to the second stage. In the second stage, a novel evolutionary-based feature selection algorithm called Tabu Asexual Genetic Algorithm (TAGA) is developed and applied on the reduced datasets. TAGA is a string type long-term memory Tabu Search (TS) hybridised with an integer-encoded asexual genetic algorithm [12], as the local search, to provide new search directions for the algorithm. Moreover, a Sequential Forward Selection (SFS) procedure is also added to enhance the performance of the algorithm. The asexual genetic algorithm is based only on a mutation search operator and lacks a crossover search operator. Crossover search operators are computationally expensive, and their absence enables the asexual genetic algorithm to efficiently guide the search process in finding high quality solutions [13]. Finally, the information theory-based minimum redundancy maximum relevance (mRMR) criterion [14] is used as the fitness function of TAGA (rather than the output of a classifier which is used in filter/wrapper methods) to evaluate the feature subsets. In this way, not only the generalisation power of the proposed algorithm is increased but also the possibility of selecting correlated features in the same subset is reduced.

The paper is structured as follows. In Section 2 related works are discussed. Section 3 describes the proposed feature selection algorithm. Section 4 explains the experimental setup including the description of datasets and competing algorithms. Section 5 discusses the experiment results. Finally, Section 6 provides the conclusion and future work.

2. Related work

2.1. Hybrid feature selection approaches

In a hybrid approach, two or more feature selection algorithms are sequentially combined which are usually of different conceptual origins [2]. Although in theory, combining two feature selection algorithms from the same type (e.g. filter/filter) is practical, the proposed approaches in the literature have mainly focused on the combination of filter and wrapper approaches (e.g. filter/wrapper) [15,8,16,17]. Filter/wrapper approaches accelerate the feature selection process and benefit from the advantages of both filter and wrapper approaches [18]. However, wrapper approaches lack generalisation power and the selected subset is often biased towards the classifier used. Therefore, the selected feature subset may not be suitable

when modifying the classifier which has been embedded in the wrapper approach, and consequently, the feature selection process will need to be repeated. From a pattern recognition perspective, feature selection algorithms should not only concentrate on classification performance, but also on finding stable and robust subsets [2].

2.2. Solution representation for feature selection in GA

The dominant GA solution representation in the literature is binary string [19]. Accordingly, binary search operators have been proposed to steer the search process. Many different approaches have been proposed to improve the performance of the GA in terms of solution representation and search operators [20–22]. Binary representation seems to be suitable for GAs embedded into wrapper approaches. This is because in a wrapper approach 2^N (N is the total number of features) subsets need to be explored and evaluated, and a binary representation allows searching through the space of all possible feature subsets. Nevertheless, when a statistical metric (such as the mRMR described in Section 3.1) is used as a fitness function, the strategy becomes intrinsically different from wrapper approaches. In this case, admissible ranges of subset cardinality (from 1 to N) are explored, and the best subset of each cardinality is evaluated over various classifiers to find the optimal subset cardinality for each classifier [4].

Integer-encoded solution representations have also been studied for feature selection. Jeong et al. [23] and Ludwig et al. [24] similarly suggested a new crossover-based GA with an integer-encoded solution representation in which the length of each chromosome is equal to the number of desired features. In cases where the index of a feature appears more than once, an SFS operator is used to find alternative features. A potential limitation of these algorithms is that they are both based on a crossover search operator and suffer from the absence of an effective mutation search operator which is important in generating diverse solutions. Because most of the feature indexes are not present in the solutions, designing a mutation operator based on swaps is difficult and, consequently, the mutation is replaced with an SFS-like procedure to repair faulty solutions. Therefore, mutation is unable to provide the search process with diverse solutions. Consequently, to address the limitation of the existing integer-encoded representation for GA, a novel representation that can reduce the dimensionality of the search space and new genetic operators that can effectively steer the search process are required.

2.3. Tabu search-based methods for feature selection

Tabu Search (TS) [25] prevents an embedded local search procedure from returning to recently visited areas (i.e. cycling) using a tabu list (TL). TLs used in TS can roughly be divided into three categories: short-term, intermediate-term, long-term. Short-term TL and intermediate-term TL have been applied in different feature selection tasks [26–28]. One drawback of using short-term and intermediate-term memory TLs is that their length needs to be carefully tuned to avoid the search process getting trapped into local optima, but only limited research exists on how to properly determine their length [29]. Therefore, a long-term memory TS can be a straightforward solution to the issues caused by short-term TS and intermediate-term memory TS, because it generates more diverse solutions by recording the entire solutions (or moves) that are generated during the search process [30]. However, long-term memory TL may require high computational time because the TL length is set to infinite [27]. Consequently, the TL becomes huge during the search process and determining whether a specific solution (or move) exists in the TL is a time consuming task.

Applications of long-term TS in the context of feature selection are limited. Wang et al. [27] developed a hybrid feature selection approach using a long-term memory TS and probabilistic neural networks. In their approach, the TL length is set to infinite and the best solution in each iteration is added to the TL. Their results on various datasets show the superiority of their approach in terms of performance compared to other works. However, their experimental results show that the running time of their approach on small datasets is very high, and this makes the algorithm infeasible for high-dimensional data. This is because the solutions are stored in the TL in their original binary representation. To address the aforementioned issue, one approach is to store the solutions in the TL using an effective encoding scheme that will accelerate the storing and restoring process.

2.4. Summary

GAs suffer from certain limitations when utilised for high-dimensional feature selection tasks. GAs require high computational time because they evaluate many solutions at each iteration [19]. To address the computational time limitation of GAs, one approach is to embed a GA into a filter/wrapper approach [19]. However, filter/wrappers use a classifier to evaluate the subsets and, therefore the selected subset is biased towards the utilised classifier. To address this limitation, a filter/filter approach is proposed which combines two filter methods. Because filter methods employ statistical metrics to evaluate the subsets, their selection process is independent of any classifier and the selected subset is unbiased towards a specific classifier. Nonetheless, using statistical metrics as the fitness functions of GAs requires an integer-encoded solution representation which allows the exploration of a range of subset cardinalities. However, binary representation is the dominant GA representation for feature selection. Consequently, this paper proposes a new integer-encoded representation and search operators. A long-term memory TL can effectively provide more diverse solutions without revisiting the same search regions. However, a long-term memory TL may require high computational time [27], particularly when the entire generated solu-

tions during the search process need to be stored in the TL. To overcome this limitation, this paper proposes a new encoding scheme for TLs which significantly improves the computational time required for storing and restoring solutions in the TL.

3. Proposed Tabu Asexual Genetic Algorithm (TAGA) for feature selection

This section presents the proposed TAGA and its components.

3.1. Minimum redundancy maximum relevance (mRMR) fitness function

The features of a dataset belong to one of three different categories: strongly relevant features, weakly relevant features, and irrelevant features [31]. While the strongly relevant features must be included in the optimal subset, the weakly relevant features are not always necessary, but may become necessary for an optimal subset at certain conditions. To determine the relevance properties of the feature space, the mutual information concept is adopted [32]. Given two random variables X and Y, their mutual information $I(X;Y)$ is defined in terms of their joint probability density and marginal functions $P_{X,Y}(x,y)$, $P_X(x)$ and $P_Y(y)$ as follows:

$$I(X;Y) = \int_y \int_x P_{X,Y}(x,y) log \frac{P_{X,Y}(x,y)}{P_X(x)P_Y(y)} dxdy \tag{1}$$

To determine the information property of a feature subset the relevance and redundancy criteria are proposed [14]. In particular the relevance of a feature subset S is defined as follows:

$$Rel = \frac{1}{|S|} \sum_{x_i \in S} I(x_i;C) \tag{2}$$

where $|S|$ denotes the number of features in the subset S, and $I(x_i;C)$ is the mutual information between a target class C and the ith variable in a feature subset S. The features are selected such that the relevance Rel is maximised, and thus it is possible to have high dependency (i.e., redundancy) amongst the selected features.

Furthermore, given two highly dependent features, and removing one of them from set S, it would not change the class-discriminative power. Hence, the redundancy of a feature subset S is defined as follows:

$$Red = \frac{1}{|S|^2} \sum_{x_i,x_j \in S} I(x_i;x_j), \tag{3}$$

where $I(x_i;x_j)$ indicates the mutual information between the ith and jth features in subset S. The purpose of feature selection therefore, is to find a feature subset S with N features that jointly have the largest dependency on the target class C and at the same time the minimal redundancy amongst themselves [14]. Consequently, this leads to a bi-criterion feature subset selection objective which is known as the minimum redundancy maximum relevance (mRMR) criterion and defined as follows:

$$mRMR = Rel - Red \tag{4}$$

3.2. Proposed solution representation

Since a statistical metric, i.e., mRMR, is used as the fitness function in TAGA, an integer-encoded solution representation appears to be more suitable [4]. Therefore, an integer-encoded representation is proposed which enables TAGA to produce subsets for a range of cardinalities as follows. Given dataset D with N features and subset cardinality P ($1 \leqslant P < N$), each feature is assigned a unique ID from 1 to N. To produce solutions, random vectors composed of all N IDs are generated and the P first features of each vector are selected. Fig. 1 presents one possible TAGA solution for a dataset composed of 10 features, where the numbers in grey correspond to the IDs of selected features. Then, the selected subset is evaluated using the mRMR fitness function described in Section 3.1.

3.3. Proposed heuristic mutation operator

A heuristic mutation search operator for the proposed solution representation is designed to swap features from the unselected part of the solution with another feature from the selected part of the solution. Fig. 2 indicates this procedure for the dataset example represented in Fig. 1. To utilise the mutual information between the target and features, two different types of feature swaps are considered. In the first type, one feature is randomly chosen from either part of the solution (i.e. selected and unselected part), and then those two features are swapped. In the second type, the feature with the lowest mutual information value between the feature and the target from the selected part of the solution is swapped with the feature with the highest mutual information value between the feature and target from the unselected part. The heuristic mutation is described in Algorithm 1.

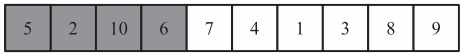

Fig. 1. Solution representation used in TAGA. The grey boxes contain the IDs of the selected features, whereas the white boxes contain the IDs of the unselected features.

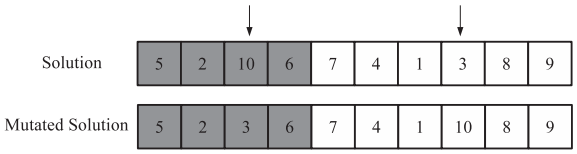

Fig. 2. Heuristic mutation operator used in TAGA. One unselected feature (e.g., 3) is swapped with one selected feature (e.g., 10). The feature IDs in grey boxes indicate the selected features whereas the feature IDs in white boxes indicate the unselected features.

It is worth mentioning that the heuristic mutation (which performs only relevance-based swaps) gives highly relevant features another chance to remain in the final feature subset. When the heuristic mutation swaps two features, one highly relevant feature is added to the subset and the subset is passed to the fitness function. With mRMR used as the fitness function, the subset is evaluated based on the relevance and redundancy values of the entire feature subset. Therefore, if the newly added feature in conjunction with other features has improved the quality of the subset, then that subset will be part of the next generation.

Note that experiments were carried out with a two-point crossover operator integrated within TAGA however, due to the poor performance of the operator, the crossover was removed from TAGA and hence, only the mutation operator was used (see Section 5.1.4 for more details).

Algorithm 1: Outline of the proposed heuristic mutation operator

```
1  begin
2      with equal probability, randomly choose a swap criterion (random or mutual
         information-based)
3      if a mutual information-based criterion is chosen then
4          find the feature with the lowest mutual information with the target from the selected
             features
5          find the feature with the highest mutual information with the target from the unselected
             features.
6          swap the features
7      else
8          randomly select a feature from the selected features
9          randomly select a feature from the unselected features
10         swap the features
11     end
12 end
```

3.4. Tabu list design

The TL in TAGA consists of a list of previous moves (i.e., swaps performed by the mutation operator described in Section 3.3) that must be avoided. Three main considerations are taken into account while designing the proposed TL, including the length of the list, the data storing strategy, and the encoding scheme.

Since a long-term memory TL is employed within TAGA, the length of the designed TL is set to infinite to store all the moves. For the second consideration, a commonly used data structure in combinatorial optimisation problems is to store the moves used to obtain a new solution [33]. For example, when a variable (e.g., a city for the well-known travelling salesman problem) or a set of variables of a solution are swapped with other ones, the swapped variables are stored in the TL. Unlike combinatorial optimisation problems for which the order the variables in the solution significantly influences the solution quality, the order of features is not important for feature selection problems, and thus, different combinations of the same variables (i.e., feature IDs in feature selection problems) lead to the same solution quality. Therefore, storing the moves used to obtain a new solution in the TL is not an effective approach for the feature selection problem using the solution representation of TAGA. For this reason, all the selected feature IDs (grey part of Fig. 1) are stored in the TL for TAGA.

For the third consideration, a new encoding scheme is proposed to overcome the existing limitations of long-term memory TLs described in Section 2.3. In particular, the selected feature IDs are separated from the original solution and are sorted in ascending order. Recall that the order of the selected features IDs is not important in feature selection problems, and thus,

sorting feature IDs in the subsets facilitates the identification of tabued solutions when the TL becomes very large. Next, the format of the ordered subset is transformed into string format by placing the feature IDs one after the other. For example, the solution represented in Fig. 1 is encoded as '25610' (note that the quotation mark indicates that the solution is represented in string format). Transforming the subset into string format is beneficial because it reduces the dimension of the TL. Given the subset cardinality m and TL length l, storing the subsets as the vectors of feature IDs requires $m \times l$ checks to determine whether a subset is tabued. However, the checks will decrease to l with the proposed string encoding scheme as each encoded tabued solution is a single set of characters.

Apart from checking whether the newly generated solutions are already stored in the TL as described above, the proposed encoding scheme employs the sorting strategy to avoid ambiguity in decoding subsets. Suppose that there are two subsets [12,1,3,1,2,13] of the same cardinality (i.e., 3), the proposed encoding scheme will convert the former to '1312' and the latter to '1213'. Note that the subsets cardinalities in TAGA are explored separately, and, hence, subsets of different cardinalities will never be sorted at the same time during the search process. Therefore, even though the encoded subset, say, '1312', can be both subsets [1,3,12,1], meaning that the encoding scheme fails to avoid ambiguity in decoding subsets, these two subsets are of different cardinalities and, thus, they will be explored separately. However, in rare cases, i.e., when the subset cardinality is very small, the encoding scheme may fail to avoid ambiguity. Suppose that there are two subsets [12,13,1] of the same cardinality (i.e., 2). The proposed encoding scheme will convert both subsets into the same subset, i.e., '1213'. In such cases, a more sophisticated encoding scheme is required (e.g., hexadecimal format which converts the two subsets to 'CD' and '1D5' respectively), which could be more concise than the decimal encoding scheme. Nevertheless, as the subset cardinality increases these rare cases are unlikely to occur.

3.5. Framework of TAGA for feature selection

Algorithm 2 provides the pseudocode of TAGA. Let D be an $m \times n$ matrix where m is the total number of records and n is the total number of features. Let N_f be the number of features selected by the Fisher score ranking algorithm [9–11] and $\mathbb{C}$ be the range of the subset cardinalities (size) from 1 to c to be explored. Let Pop_{size} and μ_r be the population size and the mutation rate of TAGA. Let $MI_{xy} = \varnothing$ be a $1 \times n$ matrix and $MI_{xx} = \varnothing$ be an $n \times n$ matrix containing mutual information between the features and the target, and pairs of features, respectively.

In the first stage of Algorithm 2 (line 1), the Fisher score feature ranking algorithm is applied to dataset D, and N_f elite features are selected to form the reduced dataset (line 2). To calculate the Fisher score, let μ_k^j and σ_k^j be the mean and deviation of the samples of the kth class, corresponding to the jth feature, and n_k be the size of the kth class. Let μ^j and σ^j denote, respectively, the mean and deviation corresponding to the jth feature. Then, the Fisher score of the jth feature is obtained by function $f(j)$ defined as follows:

$$f(j) = \frac{\sum_{k=1}^{c} n_k (\mu_k^j - \mu^j)^2}{(\sigma^j)^2}, \tag{5}$$

where c is the total number of classes and $(\sigma^j)^2 = \sum_{k=1}^{c} n_k (\sigma_k^j)^2$. After obtaining the Fisher score for all features, the N_f first features with the highest scores are selected to construct the final reduced feature subset.

Before feeding the reduced dataset to the second stage, the initial population of individuals (or solutions) in TAGA is generated (line 3), consisting of random vectors composed of integer numbers from 1 to N_f (i.e., total number of features in the reduced dataset). In the second stage (line 4), a range of subset cardinalities (i.e., number of selected features to be explored) from 1 feature to c features are explored to find the best subset for each cardinality. For this process, the selected features in the representation of individuals, N_{sel}, (grey part in Fig. 1), are set to c (line 5). Then, the number of individuals to be mutated is calculated as follows (line 6):

$$N_{mut} = (\mu_r \times Pop_{size} \times N_{sel})/2 \tag{6}$$

where Pop_{size} is population size, μ_r stands for mutation rate, and N_{sel} is the number of selected features (see Fig. 1) (i.e. cardinality).

In the next step, the N_{sel} first features of each individual are specified to be the selected features (lines 7 and 8). Line 9 evaluates the fitness of the individuals using Eq. (4). In lines 10 and 11, the necessary updates for MI_{xy} and MI_{xx} matrices, and the TL are performed. The mutual information values that are used to update these matrices are only calculated when necessary which helps in reducing the computational cost. This is because TAGA, as an evolutionary algorithm, explores particular regions (usually the regions that contain the high-quality solutions) of the search space, and some values of mutual information may never be used during the search process.

Thereafter, the proposed mutation operator (Algorithm 1) is executed N_{mut} times to generate new solutions. The new solutions which are not stored in the TL, are evaluated using the mRMR fitness function and replaced with the least-fit individuals of the population. The MI_{xy} and MI_{xx} matrices, and the TL are updated again whenever necessary (lines 13-28). This process continues until the stop criterion is reached. Finally, the best subset for each cardinality is saved for evaluations with the classifier (lines 29-31). The evolved population used in the search process for the previous subset cardinality is not discarded, but it is preserved for the search process of the next subset cardinality, which contains one additional feature. Before

the algorithm starts searching for the next subset cardinality, SFS is applied to the selected features of each individual in the evolved population to find the next suitable feature (line 32). However, in contrast to SFS where the features cannot be replaced once they are selected, in TAGA none of the features are guaranteed to remain in the final subset because during the search process these features may be replaced with higher quality features.

Algorithm 2: Pseudocode of TAGA

input : D labelled dataset, N_f number of filter stage features, $\mathbb{C}$ cardinality range, Pop_{size} population size, μ_r mutation rate, $MI_{xy} = \emptyset$ mutual information matrix between features and target, $MI_{xx} = \emptyset$ mutual information matrix between pairs of features, FEN total number of function evaluations

output: The best feature set for each cardinality

1 Apply filter algorithm (Fisher score)
2 Select N_f first features
3 Generate Pop_{size} random vectors including 1 to N_f as initial population
4 **for** each cardinality c in $\mathbb{C}$ **do**
5 Set N_{sel} to c ▷ number of features to be selected
6 Set N_{mut}=$(\mu_r \times Pop_{size} \times N_{sel}) \div 2$ ▷ number of genes to be mutated
7 **for** each individual in the population **do**
8 Set the first N_{sel} features as the selected features
9 Calculate the fitness of each individual based on selected features using mRMR ▷ Equation 4
10 Update MI_{xy} and MI_{xx} matrices
11 Update tabu list
12 **end**
13 **while** $FEN \geq 0$ **do**
14 **for** N_{mut} **do**
15 Randomly select a solution from population with a tendency to fitter individuals
16 Mutate the solution and generate a new solution ▷ Algorithm 1
17 **if** the new solution is not stored in the tabu list **then**
18 Calculate the fitness using mRMR ▷ Equation 4
19 Update MI_{xy} and MI_{xx} matrices
20 Update tabu list
21 Add the solution into new solution pool
22 $FEN = FEN$ - 1
23 **else**
24 Dispose the solution
25 **end**
26 **end**
27 Replace least-fit individuals in the population with fitter solutions in new solution pool
28 **end**
29 Sort individuals in the population in descending order based on their fitness
30 Find the best individual
31 Save the selected features of the best individual
32 Apply SFS on population to find the next suitable feature for each individual
33 **end**

4. Experimental design

4.1. Classifiers and feature selection algorithms used in the experiments

Commonly, the goodness of feature subsets is evaluated through the performance of one specific classifier, known as goal-dependent evaluation [4]. The goal-dependant evaluation, however, cannot evaluate the generalisation power of the feature subsets. The aim of these experiments is to extend the goal-independent evaluation proposed by Naghibi et al. [4], which is to compare the performance of feature selection algorithms over several datasets using multiple classifiers. This will enable the evaluation of the generalisation power.

In this paper, the goodness of the selected subsets are evaluated using five classifiers, namely the Support Vector Machine (SVM), k nearest neighbour (kNN), Classification and Regression Tree (CART), Naïve Bayes (NB), and Linear Discriminant

Analysis (LDA). This paper uses the mRMR statistical measure to evaluate the goodness of the subsets and proposes a feature selection process that is independent of any classifier. Therefore, the parameters of the classifiers will not affect the feature selection process. The parameter settings of the classifiers are provided in Table 1 to allow reproducibility of the experimental results. With regards to the k nearest neighbour (kNN) classifier, the value of k was set to 5 nearest neighbours (i.e. $k = 5$), and the distance measure was set to Euclidean, for all datasets.

The proposed TAGA[1] is compared to the following greedy search and mRMR-based algorithms:

- Sequential Forward Selection (SFS): starts from an empty set and sequentially adds the feature, that maximises the objective function when combined with the other features found in the pre-selected set of features.
- Backward Elimination (BE): unlike SFS, BE starts from the full set of features and sequentially removes the least significant feature at each iteration which improves the performance of the model. This process is repeated until no improvement is observed on the removal of features.
- ReliefF [34]: is the multi-class version of the original Relief algorithm [35], which scores the features based on the identification of feature value differences between nearest neighbour instance pairs. The feature scores are ranked and the top-scoring features are selected.
- Fisher score: scores the features such that the distances between samples from different classes become as large as possible, while the distances between samples from the same class become as small as possible [36].
- mRMR mutual information difference (mRMR-mid) [14]: is an optimal first-order incremental feature selection algorithm. It is a two-stage feature selection algorithm combining mRMR and other more sophisticated feature selection algorithms (e.g., wrappers).
- Quadratic Programming-based Feature Selection (QPFS) [37]: is based on optimising a quadratic function which is reformulated in a lower-dimensional space using the Nyström approximation method.
- Spectral relaxation Conditional Mutual Information (SPECCMI) [38]: is a global mutual information-based feature selection algorithm, in which the quadratic optimisation problem is formulated based on conditional mutual information as well as information theoretic criteria of relevance and redundancy. The formulated quadratic optimisation problem is solved via spectral relaxation.
- Customised Genetic Algorithm (CGA) [24]: searches for the best subset of features within a range of subset cardinalities using the mRMR criterion. Unlike most of the GAs in the context of feature selection which use a binary solution representation, CGA has an integer-encoded solution representation. The main algorithmic differences between CGA and the proposed TAGA are as follows:
  - CGA calculates all mutual information values at the initial stage of its execution, while in TAGA the mutual information matrices are initialised with zero values and when mutual information of new pairs of features needs to be calculated, the matrices are updated.
  - In CGA each solution is the final subset and only contains the selected features. In TAGA each solution is composed of a selected feature part and an unselected feature part in which the unselected feature part is used in the mutation search operator.
  - CGA lacks a mutation operator and needs a larger population to compensate for this deficiency. TAGA lacks a crossover operator but utilises a newly designed mutation operator which derives the search process.
  - TAGA benefits from a TL which avoids the algorithm to get trapped into local optima, while CGA does not has such a feature.
  - In CGA, an SFS algorithm is used to repair infeasible solutions generated by the crossover operator. TAGA does not need such repair function and uses the SFS algorithm to determine the next best candidate for the next subset cardinality to be explored.

4.2. Datasets and parameter settings

Table 2 shows the properties of the 10 datasets used in the experiments. All the datasets are available on the Arizona State University (ASU) feature selection repository [39], except the Dexter and DBWorld e-mails datasets that are available on the University California Irvine (UCI) machine learning repository [40]. The datasets have been widely used in previous feature selection studies and include image, text, and biological data.

One of the most important issues associated with evolutionary-based feature selection algorithms for high-dimensional datasets is that these algorithms have a high computational cost since they usually involve a large number of evaluations [19]. To resolve the computational cost issue, a popular approach is to employ a filtering stage to select the elite features to be input into the evolutionary algorithms. Therefore, the Fisher score algorithm, which is computationally cost effective, is applied to reduce the number of input features. The performance of Fisher score and its robustness to data containing noisy features for different applications have been widely discussed in the literature [10,41,42]. Note that, the Fisher score algorithm can be replaced with other algorithms.

[1] TAGA is available for download fromhttps://github.com/gcosma/TAGA/

Table 1
Parameter settings of classifiers for each dataset.

Dataset	Classifier		
	LDA discrimination type	NB data distribution	SVM kernel function
CLN	Diaglinear	Multivariate multinomial	Linear
GLI	Linear	Normal	Linear
NCI	Diaglinear	Multivariate multinomial	Linear
SMK	Linear	Kernel smoothing density estimate	Linear
TOX	Linear	Normal	Linear
LYM	Linear	Multivariate multinomial	Linear
DBE	Diaglinear	Multivariate multinomial	Linear
DEX	Diaglinear	Kernel smoothing density estimate	Gaussian
ORP	Linear	Normal	Linear
PIW	Linear	Kernel smoothing density estimate	Linear

Linear fits a multivariate normal density to each group, with a pooled estimate of covariance.
Diaglinear is similar to linear, but with a diagonal covariance matrix estimate.

Table 2
Description of the datasets used in the experiments.

Dataset	Type	# Features	# Instances	# Classes
Colon Cancer (CLN)	Biological	2000	62	2
GLI_85 (GLI)	Biological	22283	85	2
NCI9 (NCI)	Biological	9712	60	9
SMK_CAN_187 (SMK)	Biological	19993	187	2
TOX_171 (TOX)	Biological	5748	171	4
Lymphoma (LYM)	Biological	4026	96	9
DBWorld e-mails (DBE)	Text	4702	64	2
Dexter (DEX)	Text	20000	300	2
Orlraws10P (ORP)	Image	10304	100	10
Pixraw10P (PIW)	Image	10000	100	10

To determine the features to be filtered out in the first stage (i.e., size of N_f in Algorithm 2), the Fisher score algorithm ranks the features. The dataset downsizing starts by selecting the first features from the ranking list that was returned by the Fisher score algorithm, and sequentially adds more features (one by one) for creating a subset. Classification performance is obtained during the creation of the subset using the SVM, LDA, NB, *k*NN, and CART classifiers. The size of N_f for the dataset is determined when the performance of the classifiers stabilises. N_f is set to 100 features for all the datasets except for the DBE and DEX datasets, for which N_f is set to 500 features (line 3 of Algorithm 2). Afterwards, the new (i.e. reduced) datasets are generated using those elite features. For a fair comparison, the Fisher score was embedded in the filter/filter approach in the same way for the other algorithms used in the experiments.

To assess how the results of feature selection algorithms will generalise to an independent dataset, the Leave-One-Out Cross-Validation (LOOCV) was adopted. LOOCV is suitable for assessing model performance in small sample size datasets when taking into consideration model bias and estimation variance [43,44]. This paper follows the common approach for finding the optimal subset cardinality when mRMR is used as a metric criterion, which is to search for the best subsets over a range of subset cardinalities from 1 to a user-defined value [45,24,14,4]. In this paper, subset cardinalities ranged from 1 to 50, and this range was obtained experimentally. Amongst the algorithms used in the experiments, TAGA and CGA are evolutionary-based algorithms and their parameter settings are summarised in Table 3. The other algorithms are deterministic, and besides a predefined cardinality (i.e. number of desired features) they do not require further parameter tuning. In particular, the stop criterion for TAGA and CGA is the same, and these algorithms perform an equal number of function eval-

Table 3
Parameter settings of the evolutionary algorithms.

Parameter	Algorithms	
	TAGA	CGA
Population size	100	300
μ_r	0.03	–
C_r	–	0.8
Stop Criterion	$500 \times P$	$500 \times P$
Range of cardinalities $\mathbb{C}$	[1 50]	[1 50]
Tabu list length	Infinite	–

C_r and μ_r are the crossover and mutation rates respectively, and P is the subset cardinality.

uations. TAGA and CGA search for the best subsets within a range of cardinalities, and the stop criterion depends on the subset cardinality with a constant weight. The stop criterion depends on the cardinality and provides algorithms with the opportunity to search for the best subset in larger cardinalities. The constant weight of the stop criterion was investigated with $\{500, 800, 1000\}$ values and it was set to 500 ($500\times P$).

Since TAGA and CGA perform stochastic decision, it is possible to obtain different feature subsets over independent runs with different classification performances. Therefore, the algorithms were independently run 30 times for each dataset. In each run, the range of subset cardinalities is explored and the best subset found for each subset cardinality is saved and passed to 5 classifiers to gain prediction accuracy. According to these accuracy values, the best subset for each classifier is obtained. It is well known that properly tuning the parameters of the evolutionary algorithms can create a better trade-off between exploration and exploitation of the search process, and as a result, premature convergence can be prevented. For tuning the parameters of TAGA, the mutation rate and population size were investigated with values $m_r = \{0.01, 0.03, 0.05\}$ and $Pop\_size = \{100, 150, 200\}$, respectively. For tuning the parameters of CGA the crossover rate was investigated with values $C_r = \{0.7, 0.8, 0.9\}$ and the population size with values $Pop_{size} = \{100, 200, 300\}$. The parameter tuning was performed on the same set of scenarios with the same termination condition. The combination of parameters that were found to yield reasonable performance in most cases for TAGA are: $m_r = 0.03$ and $Pop\_size = 100$; and for CGA: $C_r = 0.8$ and $Pop_{size} = 300$. Note that, CGA lacks a mutation operator and needs a higher population size to compensate for the absence of a mutation operator in generating diverse solutions, and for this reason, the population size for CGA was investigated with higher values.

The average accuracy values of the best subsets over 30 runs were calculated for each classifier. The code was written in the MATLAB 2019 and Python programming languages, and experiments were performed using an Intel Core i7, 2.6 GHz processor with 64 GB of RAM.

5. Results and discussion

5.1. Experimental results of the effect of TAGA components

To examine the contribution of the proposed components in TAGA, three variations of TAGA are further defined, i.e., TAGA_{NoTabu}, $\text{TAGA}_{NoHeuristic}$ and TAGA_{NoSFS}. The first variation of TAGA (i.e. TAGA_{NoTabu}) omits the TL, but uses the proposed heuristic mutation operator (e.g., a conventional asexual genetic algorithm [12]). The second variation of TAGA (i.e. $\text{TAGA}_{NoHeuristic}$) uses the TL but the proposed heuristic mutation operator is replaced with a simple swap mutation operator. The last variation of TAGA (i.e. TAGA_{NoSFS}) omits the SFS. Table 4 presents the accuracy results of different classifiers, trained using features obtained from TAGA and the three aforementioned TAGA variations. Note that the values are averaged over 30 runs. The last column of Table 4 shows the average accuracy of the algorithms across all classifiers. For the statistical analysis, the Friedman test for multiple comparisons was applied followed by Wilcoxon signed-rank tests for pairwise comparisons (i.e. post-hoc pairwise analysis). The pairwise comparisons of TAGA against the TAGA variations averaged across all datasets are presented in Fig. 3. In the following, the effect of each TAGA component on the classification performance using five classifiers and ten benchmark datasets is discussed.

5.1.1. Effectiveness of heuristic mutation

To evaluate the performance of the proposed heuristic mutation, it needs to be compared with other mutation operators. The proposed mutation operator uses an integer-encoded representation. Most mutation operators found in the literature utilise a binary representation, and thus they cannot be compared with the proposed mutation in this paper. Therefore, the proposed heuristic mutation operator is compared with a simple swap mutation operator, in which two feature IDs are selected and swapped from any part of the solution without taking into consideration whether the chosen feature IDs are in the selected or unselected feature part of the solution.

From Table 4 and Fig. 3 it can be observed that TAGA outperforms $\text{TAGA}_{NoHeuristic}$. In the proposed solution representation, the selected subset of features (grey part in Fig. 1) contains fewer features than the unselected subset of features (white part in Fig. 1). Consequently, when a simple mutation operator is applied, it is highly possible that the two swapped features are chosen from the unselected part of the solution and, therefore, the outcome will be a repetitive final subset which is already stored in the TL. In fact, the comparisons of TAGA vs. $\text{TAGA}_{NoHeuristic}$ in Fig. 3 show that TAGA is significantly better than $\text{TAGA}_{NoHeuristic}$ in all cases, including the average accuracy case.

5.1.2. Effectiveness of SFS

To examine the effect of SFS on the performance of TAGA, the TAGA_{NoSFS} variation is compared with TAGA. From Table 4 it can be observed that TAGA, which utilises SFS, outperforms the TAGA_{NoSFS} variation. This is because TAGA explores a range of cardinalities to find the best subset. The initial population for subset cardinality of 1 (subsets with only one feature) is randomly generated and evolved during the search process. For subset cardinalities of 2 and greater, TAGA makes use of the evolved population of the previous cardinality and applies SFS to find the next suitable feature for each individual in order to generate a new population that fits the value of the next cardinality. For example, SFS is applied to the evolved population while exploring the subset cardinality of 1 and one feature is added to each of the individuals resulting in a new population

Table 4
Classification accuracy results of the different variations of TAGA over the reduced datasets. Bold values indicate best results.

Algorithm	Classifier					
	SVM	LDA	NB	*k*NN	CART	Average
CLN Dataset						
TAGA	97.4±1.3 (13.9±1.0)	90.3±0.3 (8.9±4.7)	90.3±0.0 (3.9±2.1)	94.03±0.8 (10.6±8.1)	90.1±1.6 (9.3±6.4)	**92.43**
$TAGA_{NoTabu}$	94.0±1.9 (13.3±2.1)	90.0±0.7 (8.2±5.5)	90.3±0.0 (5.3±1.1)	93.9±1.3 (5.6±4.6)	87.3±1.8 (9.8±6.3)	91.1
$TAGA_{NoHeuristic}$	89.2±1.1 (12.0±5.8)	90.2±0.5 (8.9±3.9)	90.3±0.0 (6.8±2.9)	93.4±1.4 (6.1±3.3)	86.8±1.8 (7.4±5.7)	89.97
$TAGA_{NoSFS}$	94.7±0.8 (13.4±4.6)	90.2±0.1 (8.8±2.4)	90.2±0.2 (4.8±1.3)	93.6±1.2 (8.3±1.2)	88.1±2.3 (9.4±6.1)	91.36
GLI Dataset						
TAGA	100.0±0.0 (10.6±0.9)	98.8±0.0 (10.3±0.9)	98.2±0.8 (10.1±1.6)	98.8±0.0 (14.8±4.3)	92.8±0.8 (14.4±11.9)	**97.73**
$TAGA_{NoTabu}$	99.3±1.0 (10.6±2.5)	96.5±0.6 (6.3±3.2)	97.4±0.9 (7.2±3.2)	93.2±0.7 (9.3±4.0)	92.1±1.0 (5.9±5.7)	95.69
$TAGA_{NoHeuristic}$	95.8±2.0 (6.9±1.9)	96.8±1.2 (9.0±2.3)	96.9±0.6 (7.9±2.7)	95.8±0.6 (19.1±5.6)	91.5±1.6 (15.6±10.3)	95.36
$TAGA_{NoSFS}$	98.6±0.8 (9.7±2.4)	96.8±0.8 (9.8±1.2)	97.7±0.6 (8.6±3.1)	94.8±0.7 (15.3±6.3)	92.5±0.6 (14.6±9.5)	96.08
NCI Dataset						
TAGA	84.2±1.4 (34.1±2.2)	78.1±0.5 (35.8±5.6)	83.5±0.6 (37.9±3.7)	79.5±0.4 (40.5±3.5)	60.1±1.3 (22.3±13.1)	**77.08**
$TAGA_{NoTabu}$	80.3±2.6 (35.5±6.5)	75.8±2.0 (28.5±7.3)	83.0±1.1 (42.2±7.4)	78.2±1.5 (43.1±3.5)	59.5±2.7 (26.1±16.3)	75.37
$TAGA_{NoHeuristic}$	78.5±2.8 (31.7±10.0)	76.3±1.5 (33.2±5.5)	81.8±0.5 (38.2±7.6)	77.5±1.2 (42.7±6.6)	57.5±2.9 (22.9±14.5)	74.33
$TAGA_{NoSFS}$	81.2±2.4 (34.7±6.1)	77.1±2.1 (35.2±4.3)	82.8±0.7 (40.0±5.7)	78.5±1.2 (43.6±5.6)	58.9±1.9 (25.5±12.4)	75.7
SMK Dataset						
TAGA	81.9±2.0 (19.2±4.9)	79.7±0.4 (15.1±5.0)	79.6±0.7 (9.8±5.2)	75.2±0.5 (13.1±4.4)	77.8±1.1 (16.4±4.0)	**78.88**
$TAGA_{NoTabu}$	80.2±0.8 (13.8±5.1)	79.5±0.7 (12.8±5.1)	78.4±0.4 (12.1±8.0)	74.8±1.1 (14.7±4.0)	72.7±1.1 (12.5±5.1)	77.12
$TAGA_{NoHeuristic}$	79.6±0.7 (12.2±3.3)	78.9±0.6 (12.6±3.1)	79.0±0.6 (8.6±6.2)	74.3±0.9 (14.3±1.8)	72.7±0.5 (13.4±5.7)	76.89
$TAGA_{NoSFS}$	80.4±1.1 (18.1±4.2)	78.8±1.2 (12.7±4.4)	79.1±0.8 (10.4±7.1)	74.5±1.0 (14.4±2.9)	74.6±1.5 (16.1±4.9)	77.8
TOX Dataset						
TAGA	82.4±0.7 (30.7±3.0)	81.6±1.0 (27.1±1.5)	74.9±0.8 (21.5±2.8)	77.7±0.8 (23.6±6.5)	68.4±0.9 (19.2±8.8)	**77.00**
$TAGA_{NoTabu}$	83.0±1.4 (27.2±5.8)	81.2±1.2 (27.1±2.2)	73.6±1.4 (22.4±8.5)	72.7±0.6 (27.5±10.8)	63.0±1.8 (19.8±7.9)	74.7
$TAGA_{NoHeuristic}$	81.6±1.1 (25.0±5.1)	80.1±1.0 (27.5±0.8)	72.9±0.9 (18.6±5.9)	70.8±1.0 (19.9±7.1)	63.9±2.2 (15.5±8.0)	73.85
$TAGA_{NoSFS}$	82.7±1.5 (29.1±4.5)	81.0±1.1 (27.6±1.7)	74.4±0.6 (22.9±2.1)	75.2±0.9 (24.3±7.8)	66.6±1.1 (20.1±7.1)	75.98
LYM Dataset						
TAGA	96.7±0.4 (33.5±1.5)	96.9±0.0 (40.5±0.6)	94.8±0.0 (30.6±1.8)	94.3±0.5 (20.3±3.7)	84.4±0.4 (43±2.3)	**93.42**
$TAGA_{NoTabu}$	96.9±0.0 (31.4±0.8)	93.0±1.0 (17.4±3.3)	91.9±0.4 (29.1±6.2)	93.8±0.0 (37.1±4.6)	81.3±2.1 (29.6±11.4)	91.35
$TAGA_{NoHeuristic}$	96.8±0.3 (31.1±1.2)	92.1±0.7 (18.2±3.8)	91.7±0.9 (30.2±4.6)	93.4±0.5 (31.7±4.3)	82.2±2.2 (15.2±13.4)	91.23
$TAGA_{NoSFS}$	96.8±0.5 (32.3±0.9)	94.2±1.4 (38.8±1.6)	92.2±0.5 (31.2±2.8)	93.8±0.7 (29.2±6.1)	82.9±1.3 (33.1±10.2)	91.98
DBE Dataset						
TAGA	90.6±0.0 (10.3±5.0)	90.3±1.0 (10.3±5.0)	89.1±0.0 (19.9±8.1)	90.6±0.4 (8.2±1.1)	90.3±0.7 (6.6±1.4)	**90.18**
$TAGA_{NoTabu}$	89.5±0.8 (8.5±1.8)	88.3±0.8 (10.2±5.4)	88.8±1.2 (7.0±1.6)	90.0±0.8 (6.7±1.8)	88.0±1.1 (9.6±4.4)	88.92
$TAGA_{NoHeuristic}$	88.8±1.6 (14.0±2.7)	88.4±1.5 (13.6±2.3)	88.1±1.3 (14.3±2.5)	88.8±2.2 (12.8±2.5)	87.8±1.8 (15.0±3.4)	88.38
$TAGA_{NoSFS}$	89.4±0.7 (10.3±3.4)	88.1±0.9 (11.3±4.1)	88.8±0.8 (11.4±3.1)	88.9±0.7 (9.8±1.6)	88.1±0.8 (12.5±2.6)	88.66
DEX Dataset						
TAGA	93.3±0.3 (46.7±1.5)	84.5±0.2 (39.0±2.6)	91.4±0.2 (49.3±1.6)	89.2±0.3 (37.0±6.4)	86.3±0.5 (32.7±5.7)	**88.94**
$TAGA_{NoTabu}$	93.2±0.4 (44.2±3.0)	83.7±0.2 (33.9±3.8)	90.8±0.3 (39.8±3.3)	88.2±0.4 (30.2±4.8)	86.0±0.4 (32.3±6.0)	88.38
$TAGA_{NoHeuristic}$	93.1±0.2 (47.1±1.8)	83.3±0.2 (33.1±1.7)	90.8±0.2 (42.3±1.8)	88.2±0.2 (31.2±5.0)	86.0±0.4 (26.7±9.1)	88.28
$TAGA_{NoSFS}$	93.3±0.1 (47.6±1.2)	83.6±0.3 (35.2±2.9)	90.7±0.4 (40.2±2.1)	89.0±0.1 (35.7±6.0)	86.2±1.9 (31.1±6.1)	88.56
ORP Dataset						
TAGA	97.4±0.5 (15.7±2.3)	92.3±1.2 (13.7±2.3)	94.7±0.5 (18.2±1.7)	99.8±0.4 (13.4±4.7)	86.3±2.1 (14.0±8.3)	**94.1**
$TAGA_{NoTabu}$	98.9±0.3 (13.5±1.7)	91.9±1.1 (12.2±3.4)	95.0±0.5 (18.4±1.9)	98.9±0.3 (13.8±1.0)	84.5±1.9 (13.0±8.0)	93.84
$TAGA_{NoHeuristic}$	97.2±0.4 (13.6±3.3)	91.7±1.1 (12.2±2.4)	95.0±0.0 (16.8±1.3)	99.0±0.0 (15.8±3.9)	83.7±1.8 (6.6±5.8)	93.32
$TAGA_{NoSFS}$	98.1±0.1 (13.9±3.1)	91.7±1.7 (12.3±3.1)	94.6±0.6 (18.1±1.1)	99.3±2.3 (13.8±2.8)	84.9±1.9 (12.6±7.8)	93.72
PIW Dataset						
TAGA	97.6±0.5 (6.1±1.7)	96.8±0.4 (8.0±5.4)	95.8±0.4 (15.3±6.3)	97.0±0.0 (8.1±0.7)	98.6±0.5 (9.8±4.8)	**97.16**
$TAGA_{NoTabu}$	96.9±0.3 (10.4±4.1)	97.0±0.5 (11.4±5.6)	96.2±0.6 (5.4±2.4)	97.8±0.4 (15.5±2.5)	97.2±1.0 (8.3±5.3)	97.02
$TAGA_{NoHeuristic}$	97.2±0.4 (5.7±3.0)	97.0±0.5 (10.2±6.6)	95.9±0.3 (14.7±5.9)	97.0±0.0 (7.9±2.0)	97.7±1.1 (10.7±6.4)	96.96
$TAGA_{NoSFS}$	97.0±0.5 (7.9±2.1)	96.8±0.6 (8.2±5.5)	95.6±0.5 (13.8±5.8)	96.9±0.3 (7.9±9.2)	97.9±0.3 (10.1±5.8)	96.84

The values in parenthesis present the number of selected features, and the sign ± indicates standard deviation. The last column presents the average classification accuracy results of all classifiers for each algorithm.

in which the individuals contain an additional feature. This new population is fed to TAGA as the initial population for exploring the next subset cardinality, i.e., 2. In fact, SFS guides TAGA in exploring cardinalities with a higher quality initial population as opposed to randomly generated initial population, and that is the reason TAGA is significantly better than $TAGA_{NoSFS}$ in most cases, as shown in Fig. 3.

5.1.3. Effectiveness of the tabu list

To examine the effect of the TL on the performance of TAGA, the $TAGA_{NoTabu}$ variation is compared with TAGA. From Table 4 and Fig. 3, it can be observed that TAGA, which utilises TL, significantly outperforms the $TAGA_{NoTabu}$ variation in most

	Classifier					Average
	SVM	LDA	NB	*k*NN	CART	
TAGA vs. $TAGA_{NoTabu}$	0.11	0.008	0.03	0.02	0.005	0.005
TAGA vs. $TAGA_{NoHeuristic}$	0.006	0.009	0.02	0.005	0.005	0.005
TAGA vs. $TAGA_{NoSFS}$	0.07	0.005	0.005	0.005	0.005	0.005

Fig. 3. Statistical tests of TAGA against its variations using Wilcoxon signed-rank tests on the accuracy results averaged across all datasets for each classifier. Each box contains the adjusted *p*-value of the comparison. Green boxes indicate a significant difference between the algorithms whereas grey boxes indicate no significant difference.

cases. This shows that the TL can effectively guide the search process into undiscovered areas of the search space leading TAGA to better performance.

Algorithm 3: Outline of tabu list performance analysis

```
1  begin
2      generate initial solution
3      generate multiple neighbours of the initial solution
4      set TL to ∅
5      for all neighbour solutions do
6          arrange the feature IDs in ascending order
7          if the solution is not in TL then
8              add ordered feature IDs to TL
9          else
10             ignore the solution
11         end
12     end
13     Count the number of previously visited solutions
14     Calculate the time
15 end
```

To further investigate the effect of the proposed string encoding scheme (see Section 3.4) of TL in identifying already visited solutions, the performance of the proposed TL is compared to a TL which stores solutions as vectors of feature IDs in terms of the number of correctly tabued solutions, running time, and the occupied memory space. For this purpose, Algorithm 3 is used in which an initial solution is generated using the proposed solution representation and then, a specific number of its neighbour solutions is generated. Next, each solution is checked to determine whether it had been previously generated using the competing TLs described above. For a dataset with 100 features, the initial solution is generated in such a way that 20 feature IDs are randomly selected out of 100 and then 20000 identical neighbour solutions are generated using multiple local search moves. The results are presented in Table 5, where column #UnqSol stands for the number of unique solutions generated, column #RepSol shows the number of solutions previously visited, column Time (s) indicates the running time in seconds, and the last column Space (MB) shows the memory space occupied by TLs in megabytes. As shown in Table 5, the proposed encoding scheme has correctly identified all tabued solutions. In fact, this observation supports our claim that the proposed encoding scheme may rarely fail to avoid ambiguity when small subset cardinalities are concerned, but not in large enough subset cardinalities, i.e., 20, such as the subsets considered in these experiments. Furthermore, the proposed encoding scheme has occupied almost the same memory space with the competing scheme and has performed the job almost 10 times faster. To give an idea of the practical size of an infinite list, Fig. 4 provides the number of unique solutions (#UnqSol) stored in the TL over different subset cardinality values (P) in a typical run of TAGA.

5.1.4. Effectiveness of removing the crossover operator from TAGA

To examine the effect of the crossover operator on the performance of TAGA, a two-point crossover operator was designed and embedded into TAGA. The two-point crossover operator was designed for the proposed solution representation (see Section 3.2) in such a way that two parent solutions are randomly selected from the population, and one point from the selected

Table 5
Results of the tabu list performance analysis.

Method	#UnqSol	#RepSol	Time(s)	Space(MB)
Vector of IDs	16048	3952	31.5	2.6
Encoding scheme	16048	3952	3.2	3.0

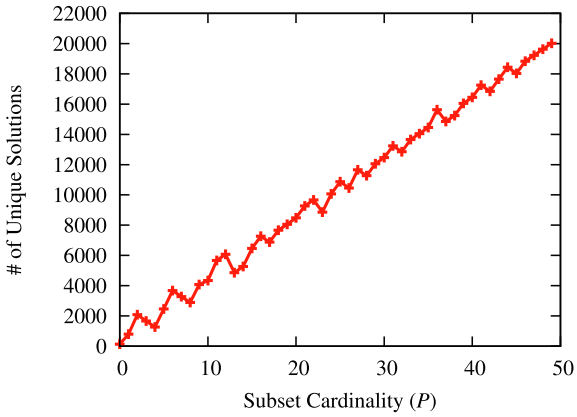

Fig. 4. Number of unique solutions generated (#UnqSol) over different subset cardinality values (P) for a typical run of TAGA on the GLI dataset.

part of each of the parents is randomly chosen. The features in between those two points are swapped to generate two new solutions. In case the generate solutions are infeasible they are repaired accordingly. The solutions generated via the designed crossover are combined with the solutions generated by the proposed mutation operator to build up the new generation. There are two strategies to combine these solutions. In the first strategy, each operator independently generates new solutions, and then newly generated solutions are merged into the population. In the second strategy, the crossover operator first generates new solutions and then new solutions along with the remaining solutions in the population are transferred to the mutation operator for further improvements.

The aim of the experiment described in this section is to select the best subset of 20 features from a sample dataset composed of 100 features. The results are provided in Table 6 in which Crossover-Mutation and Crossover → Mutation denote the first and the second aforementioned strategies, respectively. Crossover-based denotes the lack of a mutation operator, while Mutation-based denotes the existence only of a mutation operator. The typical parameter values for the crossover rate C_r and the mutation rate μ_r are provided in Table 6 (lower C_r values were investigated with no significant effect). Columns % MutRepSol and %CrossRepSol denote the percentage of the repetitive solutions generated by mutation and crossover operators, respectively. Column %TotalRepSol denotes the percentage of total repetitive solutions generated during the search process and column Time (s) indicates the running time in seconds.

From Table 6 it can be observed that the percentage of previously visited solutions (i.e., %MutRepSol) for mutation-based TAGA is lower than the percentage of the other combinations. Note that the proposed encoding scheme (analysed separately in Section 5.1.3) is not utilised in these experiments, and this was done in order to investigate the true effect of the search operators (i.e., the percentage of repetitive solutions). As a result, a simple TL, which is not involved in the searching process, is used in TAGA. Therefore, it can be inferred that mutation-based TAGA explores the search space better than the other TAGA variants that use the crossover operator. This is probably because in the crossover operator the information obtained from two parents are combined to generate a new offspring. Considering the proposed solution representation in Fig. 1, the parents are the selected feature parts of the solutions which contain a small portion of the total feature set. Therefore, the recombination of the parents, which have limited feature diversity, will most likely result in generating new solutions that have already been discovered, or which are infeasible because they contain the same feature IDs. In both cases, more computational time will be required to either cope with the repetitive solutions in TL or repair the infeasible solutions. Consequently, removing the crossover operator from the algorithm decreases the computational time, as it can be observed from the results in Table 6, without causing any negative impact on the searching capabilities of TAGA.

5.2. Comparison of TAGA with greedy search algorithms

Table 7 presents the accuracy results of different classifiers, trained using features obtained from TAGA and four greedy search algorithms (i.e. SFS, BE, Fisher score, and ReliefF). Note that the values of TAGA are averaged over 30 runs. The last

Table 6
Genetic operators performance analysis.

Algorithm	C_r	μ_r	%MutRepSol	%CrossRepSol	%TotalRepSol	Time (s)
Crossover-Mutation	0.80	0.03	6.80	78.90	53.60	5.40
Crossover → Mutation	0.80	0.03	–	–	49.80	4.30
Crossover-based	0.80	0.00	–	82.50	82.50	2.20
Mutation-based	0.00	0.03	5.60	–	5.60	2.10

Table 7
Classification accuracy results of TAGA and other greedy search algorithms over the reduced datasets. Bold values indicate best results.

Algorithm	Classifier					
	SVM	LDA	NB	*k*NN	CART	Average
CLN Dataset						
TAGA	97.4±1.3 (13.9±1.0)	90.3±0.3 (8.9±4.7)	90.3±0.0 (3.9±2.1)	94.03±0.8 (10.6±8.1)	90.1±1.6 (9.3±6.4)	**92.43**
SFS	93.5 (7.0)	90.3 (10.0)	90.3 (7.0)	91.9 (2.0)	90.3 (6.0)	91.26
BE	90.3 (8.0)	90.3 (11.0)	90.3 (8.0)	93.5 (8.0)	87.1 (2.0)	90.32
Fisher	82.3 (4.0)	79.0 (4.0)	85.5 (37.0)	85.5 (24.0)	82.3 (37.0)	82.92
ReliefF	87.1 (20.0)	88.7 (14.0)	88.7 (49.0)	96.8 (45.0)	90.3 (18.0)	90.32
GLI Dataset						
TAGA	100.0±0.0 (10.6±0.9)	98.8±0.0 (10.3±0.9)	98.2±0.8 (10.1±1.6)	98.8±0.0 (14.8±4.3)	92.8±0.8 (14.4±11.9)	**97.73**
SFS	98.8 (11.0)	98.8 (8.0)	98.8 (10.0)	98.8 (16.0)	90.6 (11.0)	97.16
BE	100.0 (9.0)	97.6 (6.0)	97.6 (9.0)	95.3 (24.0)	91.8 (10.0)	96.46
Fisher	96.5 (11.0)	94.1 (5.0)	92.9 (2.0)	89.4 (10.0)	85.9 (1.0)	91.67
ReliefF	95.3 (45.0)	90.6 (6.0)	94.1 (9.0)	92.9 (35.0)	89.4 (18.0)	92.46
NCI Dataset						
TAGA	84.2±1.4 (34.1±2.2)	78.1±0.5 (35.8±5.6)	83.5±0.6 (37.9±3.7)	79.5±0.4 (40.5±3.5)	60.1±1.3 (22.3±13.1)	**77.08**
SFS	80.0 (31.0)	75.0 (46.0)	81.7 (49.0)	76.7 (40.0)	58.3 (40.0)	74.0
BE	73.0 (34.0)	75.0 (33.0)	81.7 (45.0)	75.0 (41.0)	58.3 (8.0)	72.6
Fisher	70.0 (27.0)	66.7 (41.0)	71.7 (25.0)	73.3 (22.0)	58.3 (6.0)	68.0
ReliefF	55.0 (23.0)	53.3 (32.0)	63.3 (44.0)	53.3 (36.0)	48.3 (23.0)	54.64
SMK Dataset						
TAGA	81.9±2.0 (19.2±4.9)	79.7±0.4 (15.1±5.0)	79.6±0.7 (9.8±5.2)	75.2±0.5 (13.1±4.4)	77.8±1.1 (16.4±4.0)	**78.88**
SFS	79.1 (9.0)	80.2 (9.0)	78.6 (24.0)	75.9 (11.0)	71.7 (12.0)	77.1
BE	79.1 (37.0)	78.6 (16.0)	79.7 (38.0)	75.9 (13.0)	77.0 (17.0)	78.07
Fisher	80.2 (11.0)	79.1 (11.0)	78.6 (14.0)	73.3 (11.0)	72.7 (9.0)	76.78
ReliefF	77.5 (36.0)	72.7 (40.0)	77.0 (45.0)	70.1 (26.0)	62.6 (9.0)	71.98
TOX Dataset						
TAGA	82.4±0.7 (30.7±3.0)	81.6±1.0 (27.1±1.5)	74.9±0.8 (21.5±2.8)	77.7±0.8 (23.6±6.5)	68.4±0.9 (19.2±8.8)	77.00
SFS	78.9 (20.0)	72.5 (22.0)	75.2 (17.0)	71.3 (19.0)	64.3 (7.0)	72.44
BE	78.4 (39.0)	73.1 (11.0)	73.7 (17.0)	72.5 (32.0)	64.9 (27.0)	72.51
Fisher	83.6 (35.0)	64.3 (5.0)	66.16 (5.0)	67.8 (33.0)	64.3 (15.0)	69.22
ReliefF	98.8 (45.0)	70.8 (50.0)	81.3 (48.0)	93.6 (36.0)	65.5 (48.0)	**82.0**
LYM Dataset						
TAGA	96.7±0.4 (33.5±1.5)	96.9±0.0 (40.5±0.6)	94.8±0.0 (30.6±1.8)	94.3±0.5 (20.3±3.7)	84.4±0.4 (43±2.3)	**93.42**
SFS	94.8 (31.0)	94.8 (15.0)	93.8 (19.0)	93.8 (43.0)	82.3 (38.0)	91.9
BE	95.8 (27.0)	94.8 (31.0)	91.7 (28.0)	92.7 (31.0)	79.2 (21.0)	90.83
Fisher	91.7 (48.0)	84.4 (45.0)	83.3 (39.0)	88.5 (46.0)	82.3 (50.0)	86.04
ReliefF	94.8 (36.0)	95.8 (44.0)	95.8 (42.0)	95.8 (49.0)	79.2 (20.0)	92.28
DBE Dataset						
TAGA	90.6±0.0 (10.3±5.0)	90.3±1.0 (10.3±5.0)	89.1±0.0 (19.9±8.1)	90.6±0.4 (8.2±1.1)	90.3±0.7 (6.6±1.4)	**90.18**
SFS	87.5 (5.0)	89.1 (28.0)	89.1 (5.0)	90.6 (10.0)	89.1 (7.0)	89.08
BE	89.1 (7.0)	89.1 (32.0)	89.1 (5.0)	89.1 (7.0)	87.5 (5.0)	88.78
Fisher	68.8 (40.0)	76.6 (44.0)	73.4 (39.0)	54.7 (1.0)	68.8 (39.0)	68.46
ReliefF	92.2 (17.0)	90.6 (12.0)	90.6 (6.0)	89.1 (6.0)	85.9 (1.0)	89.68
DEX Dataset						
TAGA	93.3±0.3 (46.7±1.5)	84.5±0.2 (39.0±2.6)	91.4±0.2 (49.3±1.6)	89.2±0.3 (37.0±6.4)	86.3±0.5 (32.7±5.7)	**88.94**
SFS	93.0 (49.0)	83.3 (33.0)	90.7 (42.0)	88.3 (28.0)	86.3 (33.0)	88.32
BE	92.7 (43.0)	83.3 (3.0)	91.0 (48.0)	89.3 (49.0)	87.3 (32.0)	88.72
Fisher	83.7 (47.0)	75.0 (47.0)	72.0 (44.0)	76.0 (42.0)	77.3 (47.0)	76.8
ReliefF	87.0 (13.0)	84.3 (48.0)	73.3 (2.0)	85.3 (16.0)	84.7 (16.0)	82.92
ORP Dataset						
TAGA	97.4±0.5 (15.7±2.3)	92.3±1.2 (13.7±2.3)	94.7±0.5 (18.2±1.7)	99.8±0.4 (13.4±4.7)	86.3±2.1 (14.0±8.3)	**94.1**
SFS	97.0 (14.0)	92.0 (12.0)	94.0 (16.0)	99.0 (14.0)	85.0 (15.0)	93.4
BE	97.0 (13.0)	93.0 (13.0)	95.0 (16.0)	100.0 (45.0)	83.0 (22.0)	93.6
Fisher	94.0 (45.0)	91.0 (42.0)	87.0 (49.0)	98.0 (49.0)	78.0 (35.0)	89.6
ReliefF	81.0 (50.0)	48.0 (11.0)	66.0 (40.0)	73.0 (5.0)	69.0 (5.0)	67.4
PIW Dataset						
TAGA	97.6±0.5 (6.1±1.7)	96.8±0.4 (8.0±5.4)	95.8±0.4 (15.3±6.3)	97.0±0.0 (8.1±0.7)	98.6±0.5 (9.8±4.8)	**97.16**
SFS	97.0 (6.0)	95.0 (8.0)	95 (8.0)	97.0 (8.0)	98.0 (14.0)	96.4
BE	98.0 (6.0)	96.0 (47.0)	96.0 (21.0)	97.0 (6.0)	97.0 (19.0)	96.8
Fisher	97.0 (27.0)	97.0 (23.0)	97.0 (28.0)	97.0 (28.0)	97.0 (23.0)	97.0
ReliefF	96.0 (50.0)	85.0 (36.0)	93.0 (45.0)	94.0 (45.0)	91.0 (33.0)	91.8

The values in parenthesis present the number of selected features, and the sign ± indicates standard deviation. The last column presents the average classification accuracy results of all classifiers for each algorithm.

column of Table 7 shows the average accuracy of the algorithms across all classifiers as in Table 4 previously. For the statistical analysis, the Friedman test for multiple comparisons was applied followed by Wilcoxon signed-rank tests for pairwise comparisons. The pairwise comparisons of TAGA against the aforementioned greedy algorithms for each classifier are presented in Fig. 5e) and across all classifiers in Fig. 5(f).The *x*-axes in Fig. 5 present the dataset names and the *y*-axes present the percentage difference of the classification accuracy between the pairs of algorithms (i.e., TAGA compared with another algorithm together with the adjusted *p*-values). Comparisons with positive percentage values show that TAGA has achieved

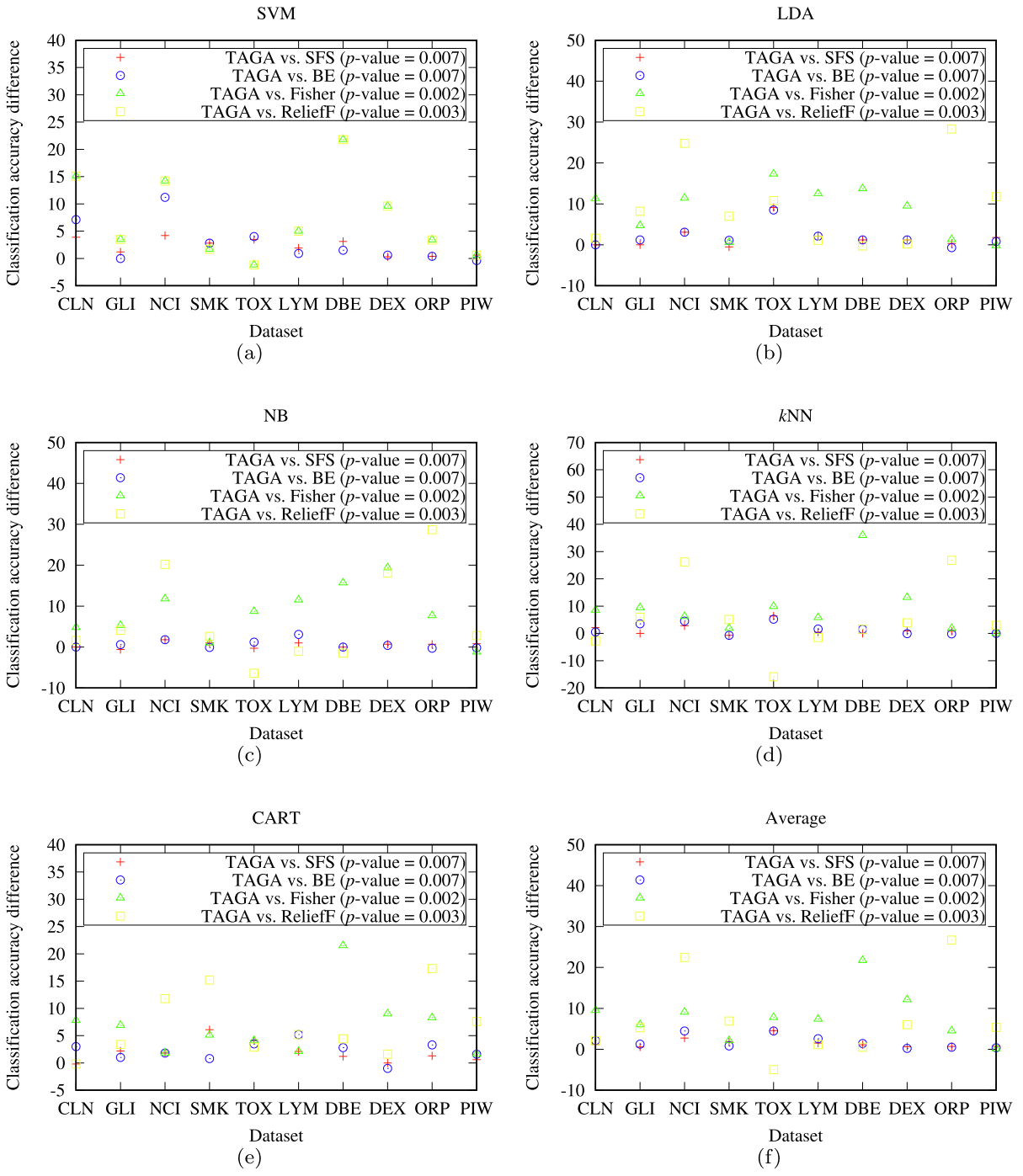

Fig. 5. Statistical comparisons of TAGA against Greedy search algorithms using the Friedman test and Wilcoxon signed-rank tests for pairwise comparisons (post-hoc analysis) applied on the classification accuracy results of each classifier (a)–(e), and averaged across all classifiers (f). Each point presents the classification accuracy difference (as a percentage) between the competing algorithms for different datasets.

higher classification performance for the corresponding datasets, while comparisons with negative percentage values show that TAGA has achieved lower classification performance than the compared algorithms. A percentage of zero shows that TAGA performs the same with the compared algorithm. For example in Fig. 5(f), the yellow point at the NCI dataset indicates that the subset selected by TAGA for the NCI dataset has achieved about 23% higher classification accuracy than the subset selected by ReliefF algorithm when considering the average classification accuracy values obtained by the five classifiers.

From Table 7 and Fig. 5, it can be observed that TAGA shows superiority over the greedy search algorithms for most cases. This is mainly because greedy algorithms make locally optimal choices to find the global optimum amongst local optima. The main disadvantage of SFS is that in each iteration of the algorithm, the usefulness of a single feature is examined in the limited context of the previously selected features only. Consequently, while selecting the final subset, the interaction between limited numbers of features is considered. In contrast to SFS, BE evaluates the contribution of a given feature in the context of all other features. However, BE overemphasises on feature interactions which may lead to a sub-optimal solution [4].

Given that Relief and Fisher score are greedy search methods, the score of each feature is computed independently. Therefore, the algorithms fail to consider the interaction between the features in a subset and as a consequence, fail to remove redundant features [36]. On the other hand, TAGA is a global search algorithm that directly searches for global optima. The embedded SFS provides TAGA with local optimal solutions that facilitate the search process (detailed analysis of the effect of SFS on the performance of TAGA is given in Section 5.1.2). However, the features in the subset provided by SFS are not guaranteed to be in the final subset as they are continuously replaced with other features for better solutions. In addition, TAGA evaluates the feature subsets (and not individual features), where each subset is composed of a portion of all features, and this prevents TAGA from overemphasising feature interactions as opposed to BE.

5.3. Comparison of TAGA with state-of-the-art feature selection algorithms

Table 8 presents the accuracy results of different classifiers, trained using features obtained from TAGA and four state-of-the-art algorithms, namely mRMR-mid [14], QFPS [37], SPECCMI [38], and CGA [24]. Note that the values of TAGA and CGA are averaged over 30 runs. The last column of Table 8 shows the average accuracy of the algorithms across all classifiers as in Table 4 previously. The same statistical analyses as described in Section 5.1 were applied. The pairwise comparisons of TAGA against the state-of-the-art algorithms averaged across all datasets are presented in Fig. 6. The x-axes in Fig. 6 present the names of the compared algorithms (together with the adjusted p-values) and the y-axes present the percentage difference of the classification accuracy between the pairs of algorithms (i.e., TAGA compared with another algorithm). Green box-plots indicate a significant difference, whereas grey box-plots indicate no significant difference. The box-plots with higher classification difference values show, that TAGA has achieved in overall higher classification performance. For example in Fig. 6(f), the corresponding box of "TAGA vs. mRMR-mid" indicates that TAGA has performed better than mRMR-mid in terms of classification accuracy, when considering the average classification accuracy values obtained by the five classifiers over the 10 datasets (positions in the last column of Table 8 and a significant difference exists in the result.

From Table 8 it can be observed that: 1) the performance of TAGA has not performed better than the other feature selection algorithms when the NB classifier was used, but it has performed better when the LDA classifier was used; 2) TAGA is the best method for datasets CLN, GLI, SMK, DBE, and ORP but not for NCI, LYM, DEX, and PIW; and 3) TAGA performs better than the other feature selection algorithms in terms of average accuracy (found in the last column of Table 8) for most datasets.

From Fig. 6, the following observations can be drawn. First, TAGA outperformed mRMR-mid in most cases. This observation was expected because the search strategy of mRMR-mid is greedy, and similar to the other greedy search algorithms discussed in Section 5.2, it searches only amongst local optima.

Second, although both QPFS and SPECCMI are quadratic programming-based algorithms, TAGA outperforms SPECCMI in most cases but it is not significantly different from QPFS. This is probably because of the different strategies the two algorithms employ Nyström algorithm to approximate mutual information. Particularly, in QPFS a two level approximation is used to cast the quadratic programming problem into a lower dimensional subspace. In this way, an acceptable approximation for small size datasets is provided but it might not yield a precise enough approximation for large datasets [4]. In contrast to QPFS, in SPECCMI, only one level of approximation with a fixed sampling rate is applied. In this way, a better approximation is provided when high redundancy exists in the dataset [38]. In fact, in the experiments of this paper before applying the feature selection algorithms, a filtering step is applied on the datasets to reduce their size, and this results in less redundancy in the datasets. Therefore, it can be concluded that the filtering step has positively affected the performance of QPFS by reducing the size of the datasets and has negatively affected the performance of SPECCMI by reducing redundancy in the datasets.

Third, although TAGA is not significantly different from QPFS, from Table 8 it can be observed that TAGA outperforms QPFS in terms of average accuracy (except in the NCI dataset). This is because QPFS approximates mutual information values using the Nyström method, and these values are less accurate than the values estimated by TAGA's mutual information estimator. Therefore, the features selected by QPFS are of lower quality than the ones selected by TAGA. The features in the NCI dataset are highly correlated and for such a dataset, lower quality features may be better for classification [4]. This may be the reason the results of QPFS are considerably better than TAGA for the NCI dataset. In fact, when NCI is removed from the statistical analysis, the performance of TAGA becomes significantly better than that of QPFS, with a p-value of 0.03.

Table 8
Classification accuracy results of TAGA and other state-of-the-art algorithms over the various reduced datasets. Bold values indicate best results.

Algorithm	Classifier					
	SVM	LDA	NB	*k*NN	CART	Average
CLN Dataset						
TAGA	97.4±1.3 (13.9±1.0)	90.3±0.3 (8.9±4.7)	90.3±0.0 (3.9±2.1)	94.03±0.8 (10.6±8.1)	90.1±1.6 (9.3±6.4)	**92.43**
mRMR-mid	90.3 (15.0)	88.7 (16.0)	90.3 (6.0)	98.4 (1.0)	90.3 (4.0)	91.61
QPFS	90.3 (2.0)	88.7 (1.0)	91.9 (7.0)	91.9 (3.0)	88.7 (1.0)	90.3
SPECCMI	90.3 (21.0)	90.3 (23.0)	90.3 (24.0)	93.5 (32.0)	87.1 (21.0)	90.3
CGA	91.5±2.6 (13.2±7.9)	90.2±0.5 (9.3±4.7)	90.8±0.8 (8.1±4.5)	95.2±1.3 (7.9±3.9)	87.4±2.0 (7.6±4.1)	91
GLI Dataset						
TAGA	100.0±0.0 (10.6±0.9)	98.8±0.0 (10.3±0.9)	98.2±0.8 (10.1±1.6)	98.8±0.0 (14.8±4.3)	92.8±0.8 (14.4±11.9)	**97.73**
mRMR-mid	96.5 (25.0)	94.1 (13.0)	95.3 (19.0)	95.3 (5.0)	95.3 (7.0)	95.29
QPFS	94.1 (8.0)	95.3 (4.0)	96.5 (5.0)	94.1 (12.0)	92.9 (23.0)	94.6
SPECCMI	98.8 (22.0)	95.3 (7.0)	97.6 (16.0)	96.5 (16.0)	90.6 (18.0)	95.8
CGA	97.8±1.0 (11.7±4.1)	96.7±1.1 (6.5±2.8)	97.2±0.6 (13.6±6.1)	95.9±0.6 (16.2±7.0)	92.0±1.8 (6.2±5.9)	95.91
NCI Dataset						
TAGA	84.2±1.4 (34.1±2.2)	78.1±0.5 (35.8±5.6)	83.5±0.6 (37.9±3.7)	79.5±0.4 (40.5±3.5)	60.1±1.3 (22.3±13.1)	77.08
mRMR-mid	66.7 (49.0)	65.0 (24.0)	65.0 (50.0)	65.0 (43.0)	58.3 (34.0)	64
QPFS	83.3 (26.0)	81.7 (14.0)	90.0 (17.0)	83.3 (39.0)	75.0 (3.0)	**82.7**
SPECCMI	85.0 (31.0)	78.3 (17.0)	83.3 (22.0)	80.0 (38.0)	68.3 (6.0)	79
CGA	83.0±2.0 (38.0±8.2)	75.7±2.6 (31.9±9.7)	79.7±2.0 (36.9±6.3)	78.8±2.4 (39.0±5.8)	59.5±3.5 (22.1±13.4)	75.33
SMK Dataset						
TAGA	81.9±2.0 (19.2±4.9)	79.7±0.4 (15.1±5.0)	79.6±0.7 (9.8±5.2)	75.2±0.5 (13.1±4.4)	77.8±1.1 (16.4±4.0)	**78.88**
mRMR-mid	77.0 (30.0)	74.3 (27.0)	74.9 (16.0)	68.4 (7.0)	76.5 (11.0)	74.22
QPFS	79.7 (18.0)	78.6 (7.0)	82.4 (16.0)	74.3 (17.0)	71.7 (6.0)	77.3
SPECCMI	77.5 (22.0)	77.5 (21.0)	79.7 (17.0)	71.1 (13.0)	70.6 (2.0)	75.3
CGA	80.4±0.8 (16.3±6.0)	79.2±0.7 (12.9±4.7)	80.1±0.7 (18.0±5.1)	72.7±1.5 (8.8±6.3)	72.8±0.9 (12.8±3.5)	77.02
TOX Dataset						
TAGA	82.4±0.7 (30.7±3.0)	81.6±1.0 (27.1±1.5)	74.9±0.8 (21.5±2.8)	77.7±0.8 (23.6±6.5)	68.4±0.9 (19.2±8.8)	**77.00**
mRMR-mid	81.3 (24.0)	78.9 (35.0)	74.3 (35.0)	72.5 (28.0)	67.8 (34.0)	74.97
QPFS	86.0 (29.0)	78.4 (14.0)	74.9 (13.0)	72.5 (15.0)	67.3 (34.0)	75.8
SPECCMI	78.4 (30.0)	77.8 (19.0)	75.4 (16.0)	77.2 (24.0)	63.7 (33.0)	74.5
CGA	82.2±1.4 (26.2±4.5)	79.8±1.0 (27.5±4.0)	73.0±1.0 (23.0±8.1)	74.3±2.6 (28.8±3.9)	64.0±2.1 (18.7±8.2)	74.64
LYM Dataset						
TAGA	96.7±0.4 (33.5±1.5)	96.9±0.0 (40.5±0.6)	94.8±0.0 (30.6±1.8)	94.3±0.5 (20.3±3.7)	84.4±0.4 (43±2.3)	93.42
mRMR-mid	95.8 (39.0)	92.7 (26.0)	92.7 (37.0)	97.9 (22.0)	85.4 (26.0)	92.92
QPFS	99.0 (26.0)	97.9 (25.0)	94.8 (15.0)	97.9 (23.0)	85.4 (22.0)	95
SPECCMI	96.9 (28.0)	94.8 (17.0)	93.8 (19.0)	93.8 (36.0)	88.5 (28.0)	**93.5**
CGA	95.7±0.8 (33.7±6.9)	93.9±1.0 (22.8±8.8)	92.8±0.8 (23.8±4.9)	94.1±1.0 (30.3±7.0)	82.7±2.5 (22.5±6.9)	91.83
DBE Dataset						
TAGA	90.6±0.0 (10.3±5.0)	90.3±1.0 (10.3±5.0)	89.1±0.0 (19.9±8.1)	90.6±0.4 (8.2±1.1)	90.3±0.7 (6.6±1.4)	**90.18**
mRMR-mid	81.3 (19.0)	85.9 (15.0)	82.8 (17.0)	89.1 (16.0)	84.4 (16.0)	84.69
QPFS	89.1 (16.0)	87.5 (10.0)	87.5 (5.0)	87.5 (13.0)	85.9 (10.0)	87.5
SPECCMI	85.9 (23.0)	89.1 (25.0)	92.2 (23.0)	87.5 (23.0)	84.4 (30.0)	87.8
CGA	88.8±1.9 (12.2±4.9)	88.4±1.3 (8.6±4.7)	93.1±1.5 (17.9±4.8)	90.6±1.3 (12.5±5.1)	87.3±1.6 (10.6±7.1)	89.66
DEX Dataset						
TAGA	93.3±0.3 (46.7±1.5)	84.5±0.2 (39.0±2.6)	91.4±0.2 (49.3±1.6)	89.2±0.3 (37.0±6.4)	86.3±0.5 (32.7±5.7)	88.94
mRMR-mid	92.7 (49.0)	84.3 (42.0)	89.7 (49.0)	88.3 (43.0)	85.0 (40.0)	88
QPFS	93.3 (49.0)	83.0 (48.0)	87.0 (48.0)	87.3 (48.0)	84.3 (30.0)	87
SPECCMI	94.0 (49.0)	84.0 (34.0)	92.3 (49.0)	89.3 (47.0)	86.3 (36.0)	**89.2**
CGA	88.1±0.8 (42.2±7.1)	76.8±0.6 (30.2±15.0)	84.0±1.3 (32.2±10.1)	82.8±1.3 (37.2±9.5)	82.3±0.6 (24.4±9.5)	82.81
ORP Dataset						
TAGA	97.4±0.5 (15.7±2.3)	92.3±1.2 (13.7±2.3)	94.7±0.5 (18.2±1.7)	99.8±0.4 (13.4±4.7)	86.3±2.1 (14.0±8.3)	**94.1**
mRMR-mid	95.0 (20.0)	86.0 (20.0)	90.0 (7.0)	97.0 (20.0)	84.0 (4.0)	90.4
QPFS	97.0 (12.0)	90.0 (12.0)	91.0 (17.0)	98.0 (9.0)	85.0 (14.0)	92.2
SPECCMI	88.0 (17.0)	77.0 (16.0)	87.0 (26.0)	94.0 (21.0)	81.0 (15.0)	85.4
CGA	97.5±0.7 (14.0±3.6)	91.4±1.6 (13.4±3.3)	94.3±1.3 (14.3±3.5)	98.5±1.0 (13.6±3.6)	84.2±2.5 (5.6±5.3)	93.18
PIW Dataset						
TAGA	97.6±0.5 (6.1±1.7)	96.8±0.4 (8.0±5.4)	95.8±0.4 (15.3±6.3)	97.0±0.0 (8.1±0.7)	98.6±0.5 (9.8±4.8)	97.16
mRMR-mid	98.0 (5.0)	99.0 (5.0)	96.0 (5.0)	96.0 (7.0)	99.0 (10.0)	97.6
QPFS	98.0 (3.0)	98.0 (20.0)	95.0 (3.0)	97.0 (20.0)	98.0 (3.0)	97.2
SPECCMI	95.0 (15.0)	97.0 (17.0)	95.0 (19.0)	95.0 (11.0)	96.0 (29.0)	95.6
CGA	97.7±0.7 (6.8±4.1)	98.4±0.7 (11.3±6.2)	96.5±0.7 (10.4±4.6)	97.2±0.4 (10.2±5.1)	98.4±0.7 (5.9±4.5)	**97.64**

The values in the parentheses present the number of selected features, and the sign ± indicates standard deviation. The last column presents the average classification accuracy results of all classifiers for each algorithm.

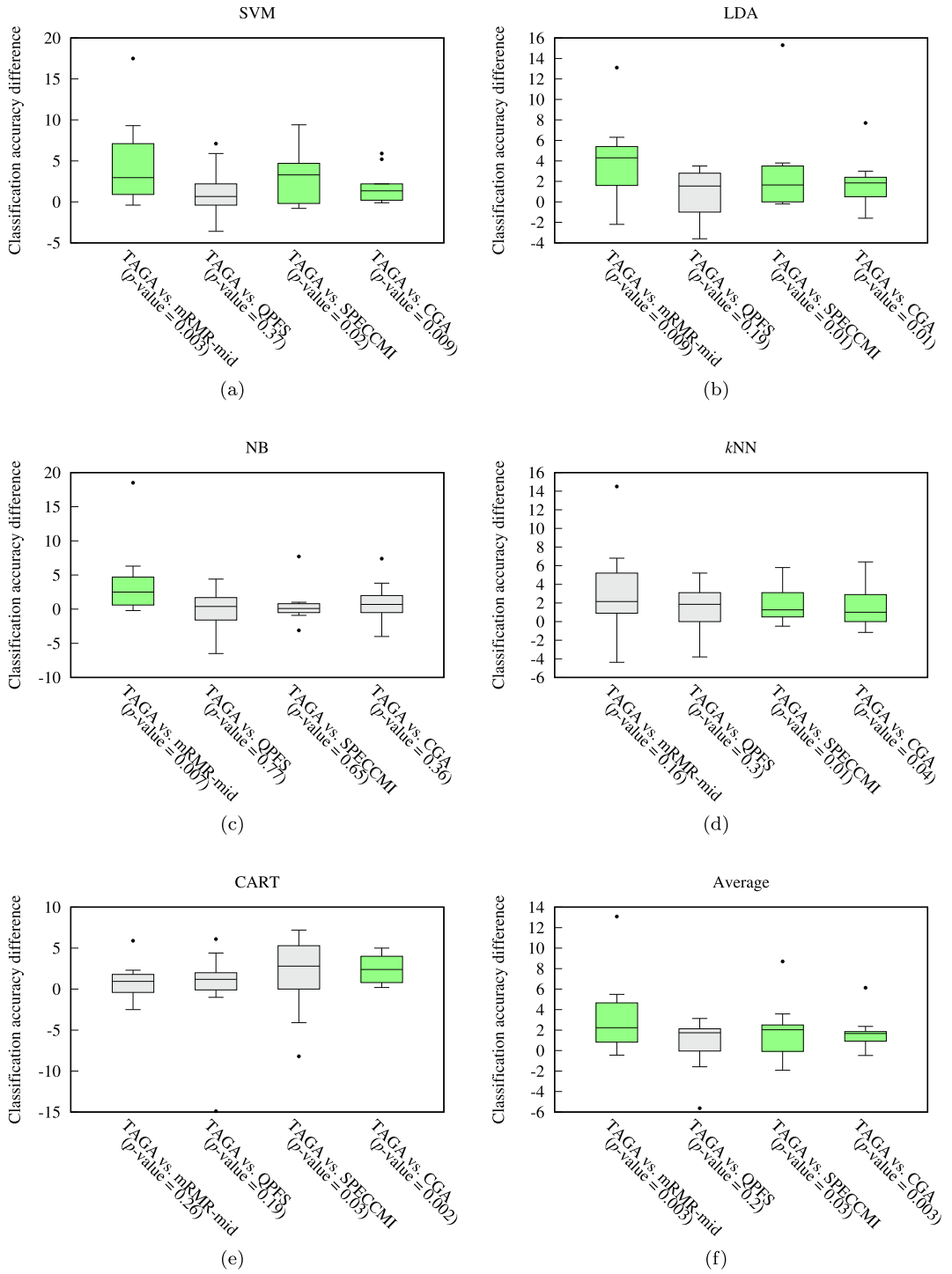

Fig. 6. Statistical comparisons of TAGA against state-of-the-art algorithms using the Friedman test and Wilcoxon signed-rank tests for pairwise comparisons (post-hoc analysis) applied on the classification accuracy results of each classifier (a)–(e), and averaged across all classifiers (f). Each box-plot presents the classification accuracy difference (as a percentage) between the compared algorithms for all datasets. Green box-plots indicate significant difference between the algorithms whereas grey box-plots indicate no significant difference.

Fourth, TAGA outperforms CGA in most cases (including the average accuracy). CGA is mainly dependent on its crossover search operator to explore the search space, and uses an SFS-like mutation operator to repair solutions. The absence of an effective mutation as the one designed for TAGA (see Fig. 2) limits the search abilities of CGA. Although both evolutionary

algorithms (i.e., TAGA and CGA) use an integer-encoded solution representation, in TAGA the solution representation contains all the features of the dataset and only part of the solution is considered as the final subset, whereas in CGA the solution representation contains the selected features only. Consequently, the mutation operator used in TAGA is not compatible with CGA. To compensate for the absence of an effective mutation operator in generating diverse solutions, the algorithm uses a large population size. In this paper, to fairly compare the two algorithms, the stop criterion is set using a predefined number of function evaluations. Hence, this fact may cause CGA to terminate at the early stages of the optimisation process due to a large number of individuals been evaluated at each generation.

5.4. Performance comparison of deep learning and conventional classifiers using TAGA selected features

This section compares the classification performance of the Self-normalising Neural Network (SNN) [46] with the classifiers used in the previous sections when using TAGA selected features. The subset of features selected from each dataset (when TAGA was run for 30 executions as described in Section 4.2) were used to train a SNN classifier and to find the cardinality at which the classifier achieves its highest classification performance.

The parameters of the SNN classifier used in the experiments are provided in Table 9. Fig. 7 shows the classification accuracy results obtained for the 10 datasets when using SNN with TAGA selected features. The same statistical analyses as described in Section 5.3 were applied. The pairwise comparisons of TAGA with SNN against TAGA with other classifiers (in Table 8) averaged across all datasets are presented in Fig. 8. The x-axis in Fig. 8 presents the names of the compared classifiers (together with the adjusted p-values) and the y-axis presents the percentage difference of the classification accuracy between the pairs of classifiers (i.e., TAGA with SNN compared to TAGA with other classifiers). Green box-plots indicate a significant difference, whereas grey box-plots indicate no significant difference.

From Fig. 8, negative classification difference values are observed in the green boxes indicating that SNN classifier is significantly outperformed by SVM, LDA, NB, and kNN classifiers, whereas it has a similar performance with CART classifier. This is mainly because SNN is specifically designed for deep learning tasks. However, the datasets used in this paper are small sample size datasets. Deep learning models typically have a very large number of nodes with many layers, and therefore many parameters need to be estimated and updated – this task requires a lot of features and samples for successful training [47]. It is anticipated, that it is for this reason that the SNN has not performed as good as the typical machine learning algorithms, which are known to perform better than deep learning methods when applied to small sample size datasets.

5.5. Generalisation power analysis

It is important to analyse the generalisation power of TAGA, and therefore how it performs over a range of classifiers. As it can be observed from Table 8, the cardinally (e.g., number of features in the optimum subset) in which the classifiers reach their highest performance (optimum subset) varies noticeably. It has been observed that the optimum subset of one classifier usually achieves optimal or near optimal accuracy, when this subset is used to train other classifiers [4]. According to this observation, Naghibi et al. [4] proposed an approach to analyse the generalisation power of a feature selection algorithm, in which the optimum subset of a classifier is used to train the other classifiers, and the results are compared to the optimal accuracy of the classifiers.

In the experiments described in this subsection, the closer the obtained accuracy values are to the classifiers' optimal accuracy values, the higher the generalisation power. Prior knowledge of optimal accuracy values of the classifiers over datasets would be useful to analyse generalisation power. However, this prior knowledge is not available. To solve the problem, the highest accuracy obtained for each classifier during the feature selection process is considered as the classifier's optimal accuracy and the corresponding subset is considered as the optimal subset. TAGA's generalisation power analysis results are shown in Table 10 using four classifiers and four datasets. The kNN optimal subset is used to train the other four classifiers.[2] As can be observed, optimal or near optimal accuracy has been obtained by most of the classifiers using the best features' subset of kNN provided by TAGA.

5.6. Running time analysis

In this subsection, the running time of TAGA is analysed. Table 11 shows the comparison of TAGA against SFS, BE, and CGA in terms of running time needed to search for the best subset over cardinality range from 1 to 50. The values provide the running time of the competing algorithms in seconds.

From Table 11 it can be observed that TAGA outperformed the other algorithms when applied to the SMK, DBE, and DEX datasets; and SFS outperformed TAGA for the datasets that have a small sample size, i.e., CLN, GLI, NCI, TOX, LYM, ORP, and PIW. In particular, a Wilcoxon signed-rank test showed that TAGA was significantly faster than BE ($Z = -2.803$, p-value = 0.005) and CGA ($Z = -2.807$, p-value = 0.005), but did not outperform SFS ($Z = -0.968$, p-value = 0.333).

TAGA achieved better running time than the other algorithms on the SMK and DEX datasets which are the largest sample size datasets; and on datasets for which more features are extracted through the first filtering stage (e.g., DBE and DEX). As

[2] The results of the remaining datasets are similar.

Table 9
Parameter settings of SNN

Parameter	Value
Activation function	SELU
Initialisation function	lecun_normal
Dropout	Alpha Dropout
Dropout rate	0.1
Hidden layers	32
Epoch	200
Batch size	5

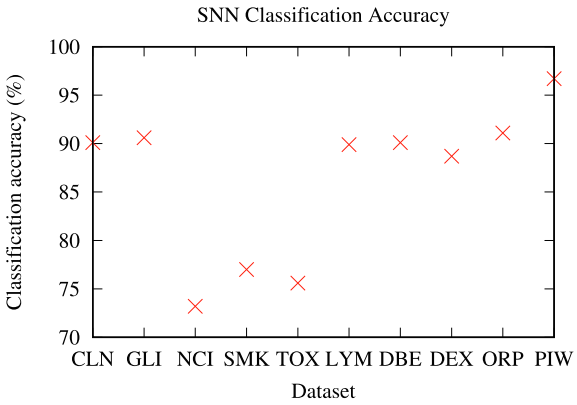

Fig. 7. Classification accuracy of the SNN deep learning classifier with TAGA selected feature subsets on different datasets.

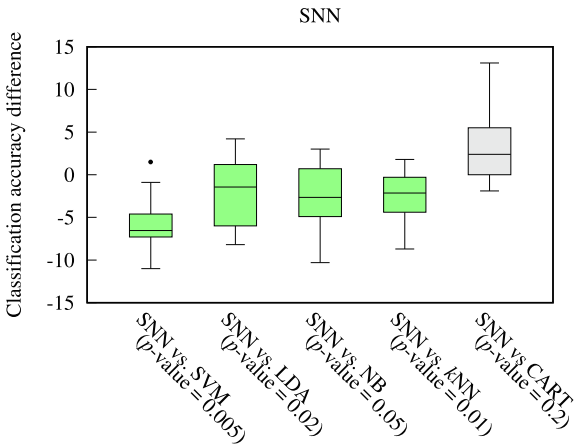

Fig. 8. Statistical comparisons of SNN against other classifiers using the Friedman test and Wilcoxon signed-rank tests for pairwise comparisons (post-hoc analysis). TAGA selected subsets are used as input to all classifiers. Each box-plot presents the classification accuracy difference (as a percentage) between the compared classifiers for all datasets. Green box-plots indicate a significant difference between the classifiers whereas grey box-plots indicate no significant difference.

previously mentioned (see Section 4.2), during the filtering stage, 100 features were filtered out for all the datasets except for the DBE and DEX datasets, for which 500 features were filtered out (line 3 of Algorithm 2).

This is probably because TAGA calculates mutual information values only when necessary, whereas SFS (which outperforms TAGA in datasets with smaller sample sizes) calculates mutual information values every time a new feature is added to the pool. Therefore, SFS needs to calculate more mutual information values between the features of larger sample size datasets, and this results in increased running time. It is worth mentioning that the Relief, Fisher, mRMR-mid, QPFS, and SPECCMI algorithms are eliminated from the comparisons. Although the Relief and the Fisher algorithms are fast, they have shown poor performance compared against TAGA (see Table 7). Regarding mRMR-mid, QPFS, and SPECCMI, these algorithms have partly been coded using various programming languages which make a fair comparison impossible. Nevertheless, the current running time analysis is sufficient to indicate TAGA's performance.

Table 10
Classification accuracy results from the generalisation power analysis of TAGA on different classifiers and datasets.

Dataset		Classifier			
		SVM	LDA	NB	CART
GLI	*k*NN optimum subset	96.47	96.44	94.12	91.76
	Classifier Optimum Acc.	100.00	98.80	98.82	94.12
DEX	*k*NN optimum subset	92.33	84.70	90.33	86.67
	Classifier Optimum Acc.	94.00	84.70	91.67	87.67
ORP	*k*NN optimum subset	97.00	92.00	95.00	82.00
	Classifier Optimum Acc.	98.00	93.00	96.00	89.00
PIW	*k*NN optimum subset	97.00	97.00	95.00	99.00
	Classifier Optimum Acc.	98.00	97.00	97.00	99.00

Table 11
Running time results (in seconds) of TAGA on different classifiers and datasets. Bold values indicate best results.

Algorithm	Dataset									
	CLN	GLI	NCI	SMK	TOX	LYM	DBE	DEX	ORP	PIW
TAGA	123	122	127	**139**	132	117	**154**	**202**	129	148
SFS	**34**	**48**	**33**	140	**123**	**59**	242	323	**62**	**63**
BE	183	223	162	649	587	294	1208	1486	288	301
CGA	158	181	168	183	174	161	229	286	199	183

6. Conclusion

This paper proposes a novel evolutionary-based feature selection algorithm called TAGA which is embedded into a new two-stage hybrid framework called filter/filter approach. The filter/filter approach was designed to address generalisation power limitations of existing filter/wrapper methods. In the first-stage, Fisher score was used to select the most informative features which were used as input into the second stage. In the second stage, TAGA which is a mutation-based GA hybridised with a long-term memory TL and guided by an SFS procedure was applied to select the final subset of features. TAGA benefits from a novel integer-encoded representation and mutation operator. In addition, a new TL encoding scheme was proposed to make the solution storing and restoring processes computationally more effective. Exhaustive experiments were performed using five classifiers and ten datasets selected from a wide range of applications to evaluate the performance of TAGA. The proposed algorithm, TAGA, was compared to greedy search and other state-of-the-art feature selection algorithms found in the literature when these were embedded in a filter/filter framework. The computational results confirmed that TAGA has outperformed other feature selection algorithms. The filter/filter approach with the embedded TAGA feature selection algorithm can be adopted when developing predictive models for biomedical or other tasks.

Future work includes improvement of the proposed method both from theoretical and application perspectives. Four future work projects are as follows. 1) The filtering algorithm and the number of filtered out features in the first stage may significantly affect the overall performance of the proposed algorithm and indeed other feature selection algorithms. Therefore, future work includes developing more efficient algorithms than filtering algorithms, for reducing the dimensionality of the original datasets. 2) To determine the optimal subset cardinality, a user-defined range of subset cardinality values is explored. Future research also includes developing methods that can automatically determine the size of the optimal subset. 3) Using the proposed string encoding scheme significantly reduces the computational time required to store and restore solutions from TL, and therefore, in future work, it is worth exploring more sophisticated encoding schemes. 4) Although a crossover operator is not utilized in TAGA because it had a negative effect, it deserves further investigation. Therefore, it would be interesting to design new crossover operators that generate feasible solutions and avoid reproducing identical solutions. 5) Finally, future work also includes evaluating the performance of the proposed TAGA and the filter/filter approach with larger and different types of datasets for specific case studies.

CRediT authorship contribution statement

Sadegh Salesi: Conceptualization, Methodology, Software, Writing - original draft. **Georgina Cosma:** Conceptualization, Supervision, Writing - review & editing, Funding acquisition. **Michalis Mavrovouniotis:** Supervision, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

Dr Cosma acknowledges the financial support of the Leverhulme Trust Research Project Grant RPG-2016-252 entitled "Novel Approaches for Constructing Optimised Multimodal Data Spaces". Dr Mavrovouniotis is partially supported by the European Union's Horizon 2020 research and innovation programme under grant agreement no. 739551 (KIOS CoE) and from the Government of the Republic of Cyprus through the Directorate General for European Programmes, Coordination and Development.

References

[1] I. Guyon, A. Elisseeff, An introduction to variable and feature selection, Journal of Machine Mearning Research 3 (2003) 1157–1182.
[2] V. Bolón-Canedo, N. Sánchez-Marono, A. Alonso-Betanzos, J.M. Benítez, F. Herrera, A review of microarray datasets and applied feature selection methods, Information Sciences 282 (2014) 111–135.
[3] R. Ruiz, J.C. Riquelme, J.S. Aguilar-Ruiz, M. García-Torres, Fast feature selection aimed at high-dimensional data via hybrid-sequential-ranked searches, Expert Systems with Applications 39 (12) (2012) 11094–11102.
[4] T. Naghibi, S. Hoffmann, B. Pfister, A semidefinite programming based search strategy for feature selection with mutual information measure, IEEE Transactions on Pattern Analysis and Machine Intelligence 37 (8) (2015) 1529–1541.
[5] G. Chandrashekar, F. Sahin, A survey on feature selection methods, Computers & Electrical Engineering 40 (1) (2014) 16–28.
[6] J. Canul-Reich, L.O. Hall, D.B. Goldgof, J.N. Korecki, S. Eschrich, Iterative feature perturbation as a gene selector for microarray data, International Journal of Pattern Recognition and Artificial Intelligence 26 (5) (2012) 1260003.
[7] A. Jović, K. Brkić, N. Bogunović, A review of feature selection methods with applications, in: Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2015 38th International Convention on, IEEE, 2015, pp. 1200–1205..
[8] H. Lu, J. Chen, K. Yan, Q. Jin, Y. Xue, Z. Gao, A hybrid feature selection algorithm for gene expression data classification, Neurocomputing 256 (2017) 56–62.
[9] R.A. Fisher, The use of multiple measurements in taxonomic problems, Annals of Eugenics 7 (2) (1936) 179–188.
[10] W. Malina, On an extended Fisher criterion for feature selection, IEEE Transactions on Pattern Analysis and Machine Intelligence 3 (1981) 611–614.
[11] R.O. Duda, P.E. Hart, Pattern Classification and Scene Analysis, Wiley, New York, 1973, pp. 114–129,221–225..
[12] J. Cantó, S. Curiel, E. Martínez-Gómez, A simple algorithm for optimization and model fitting: AGA (Asexual Genetic Algorithm), Astronomy & Astrophysics 501 (3) (2009) 1259–1268.
[13] M. Amirghasemi, R. Zamani, An effective asexual genetic algorithm for solving the job shop scheduling problem, Computers & Industrial Engineering 83 (2015) 123–138.
[14] H. Peng, F. Long, C. Ding, Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (8) (2005) 1226–1238.
[15] M.B. Dowlatshahi, V. Derhami, H. Nezamabadi-pour, A novel three-stage filter-wrapper framework for mirna subset selection in cancer classification, Informatics 5 (1) (2018) 13.
[16] E. Hancer, Differential evolution for feature selection: A fuzzy wrapper–filter approach, Soft Computing 23 (13) (2019) 5233–5248.
[17] M.M. Mafarja, S. Mirjalili, Hybrid binary ant lion optimizer with rough set and approximate entropy reducts for feature selection, Soft Computing (2018) 1–17.
[18] K.H. Hui, C.S. Ooi, M.H. Lim, M.S. Leong, S.M. Al-Obaidi, An improved wrapper-based feature selection method for machinery fault diagnosis, PLOS ONE 12 (12) (2017) 1–10.
[19] B. Xue, M. Zhang, W.N. Browne, X. Yao, A survey on evolutionary computation approaches to feature selection, IEEE Transactions on Evolutionary Computation 20 (4) (2016) 606–626.
[20] R. Li, J. Lu, Y. Zhang, T. Zhao, Dynamic adaboost learning with feature selection based on parallel genetic algorithm for image annotation, Knowledge-Based Systems 23 (3) (2010) 195–201.
[21] S.M. Winkler, M. Affenzeller, W. Jacak, H. Stekel, Identification of cancer diagnosis estimation models using evolutionary algorithms: a case study for breast cancer, melanoma, and cancer in the respiratory system, in: Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation ACM, 2011, pp. 503–510.
[22] F. Souza, T. Matias, R. Araújo, Co-evolutionary genetic multilayer perceptron for feature selection and model design, in: Emerging Technologies & Factory Automation (ETFA), 2011 IEEE 16th Conference on IEEE, 2011, pp. 1–7.
[23] Y.-S. Jeong, K.S. Shin, M.K. Jeong, An evolutionary algorithm with the partial sequential forward floating search mutation for large-scale feature selection problems, Journal of The Operational Research Society 66 (4) (2015) 529–538.
[24] O. Ludwig, U. Nunes, R. Araújo, L. Schnitman, H.A. Lepikson, Applications of information theory, genetic algorithms, and neural models to predict oil flow, Communications in Nonlinear Science and Numerical Simulation 14 (7) (2009) 2870–2885.
[25] F. Glover, Tabu search? Part I, ORSA Journal on Computing 1 (3) (1989) 190–206.
[26] E. Bonilla-Huerta, A. Hernández-Montiel, R. Morales-Caporal, M. Arjona-López, Hybrid framework using multiple-filters and an embedded approach for an efficient selection and classification of microarray data, IEEE/ACM Transactions on Computational Biology and Bioinformatics 13 (1) (2016) 12–26.
[27] Y. Wang, L. Li, J. Ni, S. Huang, Feature selection using tabu search with long-term memories and probabilistic neural networks, Pattern Recognition Letters 30 (7) (2009) 661–670.
[28] Y. Cui, J. Wang, S.B. Liu, L.G. Wang, Hyperspectral image feature reduction based on tabu search algorithm, Journal of Information Hiding and Multimedia Signal Processing (2015) 154–162.
[29] J. Dréo, A. Pétrowski, P. Siarry, E. Taillard, Metaheuristics for hard optimization: methods and case studies, Springer Science & Business Media, 2006.
[30] M. Gendreau, An introduction to tabu search, Handbook of Metaheuristics (2003) 37–54.
[31] L. Yu, H. Liu, Efficient feature selection via analysis of relevance and redundancy, Journal of Machine Learning Research 5 (Oct) (2004) 1205–1224.
[32] B.V. Bonnlander, A.S. Weigend, Selecting input variables using mutual information and nonparametric density estimation, in: Proceedings of the 1994 International Symposium on Artificial Neural Networks (ISANN?94), 1994, pp. 42–50..
[33] L. Piniganti, A survey of tabu search in combinatorial optimization, in: UNLV Theses, Dissertations, Professional Papers, and Capstones. 2132, 2014, available from: https://digitalscholarship.unlv.edu/thesesdissertations/2132..
[34] I. Kononenko, E. Šimec, M. Robnik-Šikonja, Overcoming the myopia of inductive learning algorithms with relieff, Applied Intelligence 7 (1) (1997) 39–55.
[35] K. Kira, L.A. Rendell, A practical approach to feature selection, in: Proceedings of the ninth International Workshop on Machine Learning, 1992, pp. 249–256..
[36] Q. Gu, Z. Li, J. Han, Generalized Fisher score for feature selection, in: Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence, UAI?11, AUAI Press, Arlington, Virginia, USA, 2011, pp. 266–273..
[37] I. Rodriguez-Lujan, R. Huerta, C. Elkan, C.S. Cruz, Quadratic programming feature selection, Journal of Machine Learning Research 11 (2010) 1491–1516.

[38] X.V. Nguyen, J. Chan, S. Romano, J. Bailey, Effective global approaches for mutual information based feature selection, in, in: Proceedings of the 20th ACM SIGKDD international conference on Knowledge Discovery and Data Mining, ACM, 2014, pp. 512–521.
[39] J. Li, K. Cheng, S. Wang, F. Morstatter, R.P. Trevino, J. Tang, H. Liu, Feature selection, ACM Computing Surveys 50 (6) (2018) 1–45.
[40] A. Asuncion, D. Newman, UCI machine learning repository (2007)..
[41] S. Olyaee, Z. Dashtban, M.H. Dashtban, Design and implementation of super-heterodyne nano-metrology circuits, Frontiers of Optoelectronics 6 (3) (2013) 318–326.
[42] J. Xuan, Y. Wang, Y. Dong, Y. Feng, B. Wang, J. Khan, M. Bakay, Z. Wang, L. Pachman, S. Winokur, Y.-W. Chen, R. Clarke, E. Hoffman, Gene selection for multiclass prediction by weighted Fisher criterion, EURASIP Journal on Bioinformatics and Systems Biology 2007 (2007) 64628.
[43] R. Kohavi, A study of cross-validation and bootstrap for accuracy estimation and model selection, in: Proceedings of the 14th International Joint Conference on Artificial Intelligence – Volume 2, IJCAI?95, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1995, p. 1137–1143..
[44] A.M. Molinaro, R. Simon, R.M. Pfeiffer, Prediction error estimation: a comparison of resampling methods, Bioinformatics 21 (15) (2005) 3301–3307.
[45] C. Ding, H. Peng, Minimum redundancy feature selection from microarray gene expression data, Journal of Bioinformatics and Computational Biology 3 (02) (2005) 185–205.
[46] G. Klambauer, T. Unterthiner, A. Mayr, S. Hochreiter, Self-normalizing neural networks, in, Advances in Neural Information Processing Systems (2017) 971–980.
[47] W. Zhao, Research on the deep learning of the small sample data based on transfer learning, AIP Conference Proceedings 1864 (2017) 020018.