Genetic Programming and Evolvable Machines (2023) 24:10
https://doi.org/10.1007/s10710-023-09458-y

RESEARCH

GAAMmf: genetic algorithm with aggressive mutation and decreasing feature set for feature selection

Rejer Izabela[1] · **Lorenz Krzysztof**[1]

Received: 16 September 2022 / Revised: 9 June 2023 / Accepted: 27 June 2023 /
Published online: 26 July 2023

Abstract

This paper introduces a modified version of a genetic algorithm with aggressive mutation (GAAM), one of the genetic algorithms (GAs) used for feature selection. The modification proposed in this study expands the original GAAM's capabilities by allowing not only feature selection but also feature reduction. To obtain this effect, we applied the concept of ranks used in the non-dominated sorting genetic algorithm (NSGA) and the concept of penalty term used in the Holland genetic algorithm. With those two concepts, we managed to balance the importance of two competing criteria in the GAAM fitness function: classification accuracy and the feature subset's size. To assess the algorithm's effectiveness, we evaluated it on eleven datasets with different characteristics and compared the results with eight reference methods: GAAM, Melting GAAM, Holland GA with a penalty term, NSGA-II, Correlation-based Feature Selection, Lasso, Sequential Forward Selection, and IniPG (an algorithm for particle swarm optimisation). The main conclusion drawn from this study is that the genetic algorithm with aggressive mutation and decreasing feature set (GAAMmf) introduced in this paper returned feature sets with a significantly smaller number of features than almost all reference methods. Furthermore, GAAMmf outperformed most of the methods in terms of classification accuracy (except the original GAAM). In contrast to Holland GA and NSGA-II, GAAMmf was able to perform the feature reduction task for all datasets, regardless of the initial number of features.

Keywords Feature selection · Genetic algorithm · Aggressive mutation · Holland · NSGA · GAAM

Rejer Izabela and Lorenz Krzysztof have contributed equally to this work.

Handling Editor: Sebastian Risi.

Extended author information available on the last page of the article

1 Introduction

The algorithms used for classification, regression, or clustering typically do not perform well on high-dimensional data [1]. Therefore, a traditional approach in machine learning is to implement procedures to reduce the number of features used by those algorithms. The goal of all feature selection procedures is to find a small subset of features that provide a high value of the implemented performance measure.

In many areas, the actual feature reduction rate is not a critical parameter. For example, when the algorithm's task is to identify cancer areas in medical images in offline mode, additional features may be acceptable if they enhance recognition. However, there are areas where we cannot afford any additional features. The need for the smallest possible feature set is particularly crucial when the classifier is used for real-time decision-making, and the time needed to extract each feature from the current stream of data is considerably high. In such a situation, each additional feature that must be extracted from the data stream is an unnecessary burden that can disrupt the real-time classification or even make it impossible.

There are many traditional feature selection approaches, including filters such as ReliefF [2], Correlation-based Feature Selection [3], and Consistency-based Feature Selection, as well as wrappers like step-wise selection [4, 5], random selection [6], and Recursive Feature Elimination [7]. Additionally, embedded methods such as Lasso [8] can also be used. Besides classic approaches, heuristic approaches inspired by nature are becoming increasingly popular. A comprehensive review of these approaches is provided in [9], which reports dozens of algorithms inspired by the behaviour of insects, reptiles, birds, and animals.

Among all nature-inspired feature selection algorithms, the most popular and commonly used are those based on swarm intelligence (SI) and genetic algorithms (GA). Some of the well-known algorithms from the first category are particle swarm optimization (PSO) [10–14], ant colony optimization (ACO) [15–17], and artificial bee colony optimization (ABC) [18–20]. Many studies have shown that SI algorithms are efficient feature selection techniques [21–23]. However, according to [1], most SI-based feature selection algorithms suffer from the poor scalability of representation, which usually makes them unsuitable for applications where thousands or millions of features are possible. Furthermore, although SI-inspired feature selection algorithms use different types of representation (either binary or continuous), they do not directly support the search process among feature sets containing a fixed and very small portion of all the possible features. Hence, when the smallest possible feature subset is needed, GA-based approaches might be a better option [24–31].

One advantage of using a GA as a feature selector is that it evaluates the entire set of solutions simultaneously, instead of sequentially [24]. Therefore, it can explore different parts of the search space in the same generation, rather than focusing on one particular area and potentially getting stuck in a local minimum. Additionally, even if it falls into a local minimum, it can escape on its

own. Another feature of GA that is critical for the feature selection process is that it does not assume any interactions among features existing in the feature set [32–34]. However, the main disadvantage of using GAs as feature selectors is their long processing time, which is a consequence of wrapping the feature selection process around a classification scheme [6, 35].

Several GA approaches have been proposed to solve the feature selection problem, with one of the most popular being based on the classic Holland GA [27, 31, 32, 34, 36–38]. In this approach, features are coded into genes using a binary scheme, wherein a gene possessing a value of 1 denotes that the feature is included into an individual, while a gene possessing a value of 0 indicates its absence. With this scheme, the number of genes in an individual is equal to the total number of features in the feature space. However, a high number of genes in an individual has two disadvantages. First, when the GA fitness function is based solely on classification accuracy, as in the classic Holland GA, the classifiers are trained with a massive number of features, which is a time-consuming process. Second, although the classifiers' accuracy is often approximately 100% at the very first algorithm iteration, most have limited application due to their limited generalization capabilities. Both problems can be resolved by introducing a penalty term into the classic Holland GA's fitness function, penalizing individuals with too many features [24, 39, 40]. This solution generally achieves this task, but the feature reduction process is slow since the individuals in the initial population start with approximately half of all available features.

Another well-known feature selection approach based on a genetic algorithm is the non-dominated sorting genetic algorithm (NSGA), and its modification, NSGA-II [24, 25, 40–45]. The NSGA algorithm encodes features into individuals using a binary scheme, as in the classic Holland GA. The main difference between the algorithms is the optimization criterion. While the Holland GA optimises only the classification accuracy (maximization), the NSGA algorithm optimizes two criteria simultaneously: classification accuracy (maximization) and the number of features encoded in an individual (minimization). To perform this simultaneous optimization, the NSGA fitness function uses the domination principle, where individuals are assigned different ranks based on their level of dominance. The problem with the NSGA algorithm is similar to that with the Holland GA. Both algorithms start with a random population of individuals containing approximately 50% of the possible features (a result of a binary coding scheme) and then try to find a balance between the individuals' accuracy and their number of features, which is a time-consuming process.

In [40], we proposed another GA dedicated to feature selection, a genetic algorithm with aggressive mutation (GAAM). This algorithm was designed to address one of the problems of the Holland approach, which emerged in datasets composed of thousands of features—the huge size of individuals. To overcome this issue, we changed the coding scheme from binary to integer, with genes encoding the feature indexes. We also designed a new mutation operator and adjusted other GA steps. These changes allowed us to use the algorithm with individuals of arbitrary length, from individuals containing only one gene to individuals coding all the features from the feature space. However, we encountered a subsequent problem.

The classic version of GAAM works with individuals of a fixed number of genes (features) set as a parameter. This highly restricts the possibility of training over-sized classifiers, but also limits the algorithm's ability to search for subsets of features smaller than the initially chosen size. The only chance for decreasing the number of features is when an individual with repeated features is born in the reproduction process. When such a situation occurs, one of the redundant features is discarded, and the individual shrinks. Unfortunately, this is a sporadic event. Hence, although GAAM works well in terms of accuracy, classifier generalisation capabilities, and computation time, as compared to other approaches [40, 46–49], there is still room for improvement in changing individuals' length. Although we can set the number of individuals' genes to a reasonable value for the given task, we cannot predict if this value is optimal. Therefore, from the beginning of our work with GAAM, we have tried to adjust the algorithm so it could reduce the initial number of genes. To address this problem, we proposed a second version of the algorithm: Melting GAAM [24].

Melting GAAM uses an iterative approach to reduce the number of genes in individuals. It begins with the number of genes set by the user and attempts to improve classification accuracy to the specified level (also set by the user) during a given number of iterations. If the desired level is achieved, one random gene is removed from all individuals in the current population, and the optimization process restarts. The algorithm stops when it cannot surpass the accuracy threshold within the given number of iterations. While this approach works well, it is not flawless. Firstly, the process of optimising the classification accuracy is repeated numerous times for each number of genes. Although the process restarts from individuals highly similar to those from the previous iteration (with only one gene discarded), it requires some time for the algorithm to stabilize. Secondly, setting the accuracy threshold correctly is challenging: too high threshold quickly halts the feature reduction process, while too low threshold removes many features but yields individuals with low accuracy.

In this paper, we propose an alternative solution to decrease the number of individuals' genes. The proposed solution utilizes both, the penalty term employed in Holland GA and the ranks assigned to individuals in the NSGA algorithm. The paper presents a detailed algorithm of our approach, called Genetic Algorithm with Aggressive Mutation and Minimum Features (GAAMmf), and the results of experiments comparing its performance to four other GAs (GAAM, Melting GAAM, Holland with a penalty term, and NSGA-II) and four non-genetic feature selection methods (Correlation-based Feature Selection (CFS), Lasso, Sequential Forward Selection (SFS) [4] and IniPG [11]—one of PSO algorithms) conducted on eleven datasets with different characteristics. The experiments demonstrate that GAAMmf produces individuals encoding feature sets of comparable accuracy but containing a significantly smaller number of features. Furthermore, in addition to returning the final solution, the algorithm also generates individuals with the highest accuracy for each analyzed number of features. Due to this feature, the algorithm's user can decide by himself which feature subset is preferable, larger but more precise or less precise but containing a smaller number of features.

The paper's structure is as follows. The subsequent section describes the main concepts of GAAM and provides a detailed explanation of GAAMmf. The following

two sections discuss the experimental setup used to compare the algorithm's performance and report the results. Finally, the last section concludes the paper.

2 Methods

The algorithm described in this manuscript is a modification of the GAAM algorithm introduced in [25]. The proposed modification involves altering the fitness function used to evaluate individuals in subsequent generations. While in the original GAAM, the individuals are ordered exclusively according to their classification accuracy, GAAMmf combines two criteria in the fitness function, classification accuracy and the number of features encoded in an individual. This change allows for a gradual decrease in the number of features in successive generations, which is unattainable with the original GAAM algorithm.

2.1 The original GAAM

The pseudocode of the original GAAM is presented in Algorithm 1. The algorithm begins by setting four parameters: *N*, *M*, *T*, and *tourN*. Parameters *N* and *M* represent the initial population size, where *N* denotes the number of genes (features) encoded in each individual, and *M* denotes the number of individuals in the population. Parameter *T* determines the stopping condition for the algorithm, i.e., the number of generations to perform. Lastly, parameter *tourN* indicates the number of individuals used in the tournament selection procedure.

After setting the algorithm's parameters, the initial mother population *motherP*) is drawn (function: DrawInitialPopulation). The population is composed of *M* randomly selected individuals, where each individual contains *N* genes. The genes are integer values corresponding to the indexes of features in the set of possible features *1*,..., *P*}.

Algorithm 1 The GAAM's pseudocode; N – number of genes, M – number of individuals, T – number of generations, $tourN$ – number of individuals in tournament

Set N, M, T, $tourN$
$motherP$ = DrawInitialPopulation(M)
for $generation$ = *1* ***to*** T **do**
  $crossedP$ = OnePointCrossover($motherP$)
  $mutatedP$ = AggressiveMutation($motherP$)
  $currentP$ = [$motherP$ $mutatedP$ $crossedP$]
  $fitness$ = FitnessEvaluation($currentP$)
  [$motherP$, $fitness$] = TournamentSelection($currentP$, $fitness$, $tourN$)
end
$theBestIndividual$ = MaxFitness($motherP$, $fitness$)

The main algorithm loop commences with two reproduction operations executed on the individuals from the initial population (*motherP*): a traditional one-point crossover (function: OnePointCrossover) and the aggressive mutation (function: AggressiveMutation). The latter is a GAAM-specific concept, performed individually on each gene of each individual, in accordance with the pseudocode presented in Algorithm 2. The two populations created during the reproduction operations are concatenated with the mother population, forming the final population (*finalP*).

Subsequently, each individual is assessed based on the classification accuracy attained by the classifier utilizing the features encoded in the individual's genes. Following evaluation, the *M* best individuals are selected from the current population using the tournament selection procedure. The algorithm terminates after achieving the predefined number of iterations.

Algorithm 2 The pseudocode for the AggressiveMutation function; P – the total number of features in the feature space

```
j=0
for i = 1 to M do
    newIndividual = motherP(i, 1 : N)
    for g = 1 to N do
        newIndividual(g) = Random({0, 1, ..., P})
        j++
        mutatedP(j, 1 : N)=newIndividual
    end
end
```

2.2 The GAAMmf

As mentioned previously, the primary motivation for developing a new version of GAAM was to further reduce the number of features returned by the algorithm. To achieve this goal, we designed the GAAMmf fitness function to incorporate two criteria: classification accuracy, as in the original GAAM, and the number of features encoded in an individual. This new fitness function required several modifications in the GAAMmf algorithm, which are described below and presented in pseudocode in Algorithm 3. The three functions—DrawInitialPopulation, OnePointCrossover, and AggressiveMutation—remain unchanged and perform the same operations as described in Sect. 2.1.

Algorithm 3 The GAAMmf's pseudocode; N – number of genes, M – number of individuals, T – number of generations, $probM$ – mutation probability, $accWeight$ and $fsWeight$ – weights to regulate the importance of the fitness criteria

Set N, M, T, $probM$, $accWeight$, $fsWeight$
$motherP$ = DrawInitialPopulation(M)
$motherAcc$ = accFitness($motherP$)
$accFactor$ = 100 * (100 / (N - 1))
$fsFactor$ = (N - 1) * (100 / (N - 1))
for $generation$ = *1* ***to*** T **do**
  $crossedP$ = OnePointCrossover($motherP$)
  $mutatedP$ = AggressiveMutation($motherP$)
  $childP$ = [$mutatedP$ $crossedP$]
  $childAcc$ = accFitness($childP$)
  $finalP$ = [$motherP$ $childP$]
  $finalAcc$ = [$motherAcc$ $childAcc$]
  $fitness$ = GAAMmfFitness($finalP$, $finalAcc$)
  [$motherP$, $motherAcc$] = RankSelection($finalP$, $fitness$)
end
$bestIndividuals$ = MaxFitnessPerFeatures($motherP$)

First, we had to ensure that the algorithm's individuals would be composed of a variable number of genes. We achieved this goal by using a redundant number of genes at the beginning of the algorithm. Second, we had to ensure the comparability of both criteria used in the evaluation function. To deal with this task, we employed the concept of ranks from the NSGA-II algorithm and assigned a set of ranks to different levels of accuracy and another set of ranks to different numbers of features. Third, we had to ensure that the original (unranked) values of both criteria would be passed between successive populations. To this end, we divided the algorithm evaluation function into two parts. The first part (function: accFitness) evaluated each individual's accuracy. The second part counted the individuals' features, ranked them individually according to both criteria, and calculated the final fitness of each individual (function: GAAMmfFitness). While we used the accFitness function twice in the algorithm body, first to evaluate the initial population and then in the main algorithm loop to evaluate each new population, the GAAMmfFitness function was used only once: in the main loop, after concatenating mother and child population.

Finally, to ensure an equal contribution of both criteria in the total fitness value, we added two algorithm constants, accFactor and fsFactor, which were calculated based on the number of possible accuracy levels and the number of possible features in an individual. We assumed that the accuracy levels range from 0 to 100% (100 integer levels) and the number of features, from *1* to *N* (*N*−*1* levels).

Under these assumptions, the *accFactor* and *fsFactor* constants were calculated as shown in Formulas (1) and (2), respectively.

$$accFactor = 100\frac{100}{N-1} \tag{1}$$

$$fsFactor = (N-1)\frac{100}{N-1} = 100 \tag{2}$$

In GAAMmf, the fitness of individuals is calculated within the GAAMmfFitness function (Algorithm 4). The function takes two input parameters: the final population of individuals, which includes individuals from the mother population and all off-springs born during the crossover and mutation operations, and the accuracy vector containing the classification accuracy of all individuals from the final population. Inside the function, a sequence of seven operations is performed.

First, the individuals from the final population are sorted according to their increasing accuracy. Then, they are ranked based on the rule that individuals with the same accuracy (rounded to integer values) are assigned the same rank. Rank 1 is assigned to individuals with the worst accuracy. Since the range of ranks can differ for both criteria, the ranks are normalized using a pseudo min-max normalization (3) after assigning accuracy ranks to all individuals. We refer to this as "pseudo min–max normalization" to indicate that accuracy values are normalized based on fixed boundaries set to 0 and 100 for the accuracy criterion, rather than the minimum and maximum accuracy obtained in the current population.

$$accNorm(i) = \frac{acc(i)}{100} \tag{3}$$

where *accNorm(i)*—accuracy rank of individual *i* after normalisation, and *acc(i)*—accuracy rank of individual *i*.

The three steps described for the accuracy criterion are repeated for the number of features criterion, with two subtle changes. First, since the worst rank (Rank 1) for this criterion should be assigned to individuals with the largest number of features, individuals are sorted in descending order. Second, the pseudo min–max normalization boundaries are set to *1* and *N*, indicating that the normalized ranks are calculated according to Formula (4).

$$fsNorm(i) = \frac{fs(i)-1}{N-1} \tag{4}$$

where *fsNorm(i)*—number-of-feature rank of individual *i* after normalisation, *fs(i)*—number-of-feature rank of individual *i*, and *N*—number of genes in an individual.

Finally, the normalized ranks are multiplied by the corresponding weights and added together, as shown in (5):

$$\begin{aligned} fitness(i) = {} & accWeight * accFactor * accNorm(i) \\ & + fsWeight * fsFactor * fsNorm(i) \end{aligned} \tag{5}$$

where *fitness(i)*—the final fitness of individual *i*, *accNorm(i)* and *fsNorm(i)*—the normalized ranks of individual *i* with respect to the accuracy (*accNorm*) and the number of features (*fsNorm*) criteria, *accFactor* and *fsFactor*—factors that ensure an equal contribution of both criteria to the total fitness, *accWeight* and *fsWeight*—weights that allow the importance of both criteria to be regulated.

Algorithm 4 The pseudocode for the function GAAMmfFitness

$finalP = \text{SortByAccuracy}(finalP,\ finalAcc)$
$acc = \text{rankByIntegerAccuracy}(finalP)$
$accNorm = \text{NormalizeMinMax}(acc)$
$finalP = \text{SortByFeatureNo}(finalP)$
$fs = \text{rankByFeatureNo}(finalP)$
$fsNorm = \text{NormalizeMinMax}(features)$
$fitness = accWeight{*}accFactor{*}accNorm + fsWeight{*}fsFactor{*}fsNorm$

Since the algorithm simultaneously explores subsets of features of different sizes, its output is not a single individual with the best characteristics, but a set of individuals. Each individual in the final set of *bestIndividuals* represents the solution of the highest accuracy obtained for the feature set of the given number of features. As a result, the algorithm user might decide which solution better suits their needs: that with slightly lower accuracy but a smaller number of features or that with slightly higher accuracy but a higher number of features.

Apart from the change in the fitness function, we introduced the two other subtle modifications that we had tested previously in other papers on GAAM [25, 50]. Firstly, we introduced a new algorithm parameter *probM* (probability of mutation) to control the intensity of the mutation process. This parameter enhanced the algorithm's scalability and enabled its application for problems described in a high-dimensional feature space.

Secondly, we changed the selection procedure from tournament selection to rank selection. This alteration resulted in a significant reduction of the computational burden imposed by the algorithm. By employing aggressive mutation, the GAAMmf explores various regions within the problem space during each iteration. Consequently, it requires only a limited number of iterations to attain the final results, although each of these iterations is computationally intensive. The transition from tournament selection to rank selection facilitated a decrease in the number of iterations needed (as the best individuals consistently prevail in rank selection), thereby leading to a significant reduction in the overall processing time of the algorithm.

3 Experiment setup

To evaluate the effectiveness of GAAMmf, we conducted a study using eleven datasets, namely *Pima-Indians-diabetes*, *Orlraws10P*, *Dermatology*, *Adult*, *Gisette*, *Humanactivity*, *Coil100*, *Gli_85*, *Orl_32×32*, *WarpAR10P*, and *Yale_32×32*,

downloaded from sources cited in Table 1. The datasets differed in terms of the number of features, classes, and examples. The aim of our study was to demonstrate that regardless of the dataset characteristics, GAAMmf produces individuals with classification accuracy comparable to reference methods but containing significantly fewer genes. The results of GAAMmf were compared with those of the original GAAM and three other genetic approaches that allow for changes in the number of individuals' genes, namely Melting GAAM, Holland GA with a penalty term, and NSGA-II. In addition to genetic algorithms, we compared the GAAMmf results with results returned by four non-genetic feature selection methods (CFS, Lasso, SFS, and IniPG).

Before using the datasets in the study, we applied the following preprocessing procedures: (i) removal of all records containing *NaN* values, (ii) removal of redundant features (features that had the same value for each record), and (iii) identification of pairs of features whose linear correlation exceeded 99%, and discarding one feature from each pair. The detailed demography of the datasets before and after applying the preprocessing procedures is presented in Table 1.

For all datasets, the main GAAMmf parameters were set at the same levels: M (number of individuals in the mother population) was set to 10, *probM* (mutation probability) to 1, and T (number of algorithm iterations) to 100 (for the first two experiments) or 1000 (for the last experiment). The value of the N parameter, denoting the initial number of individual's genes, was also standardized for most datasets and was set to 20. Only for two datasets, namely *Adult* and *Prima-Indians-diabetes*, which contained 14 and 8 potential features, respectively, the N parameter was set to the total number of features.

The accuracy of individuals was evaluated using a linear discriminant analysis (LDA) classifier. Our decision to employ the LDA classifier was motivated by two key factors. Firstly, the adoption of a linear classification procedure allows for the generation of a classification model with a minimal number of parameters.

Table 1 The characteristics of the datasets used in the survey

Dataset	No. of features	No. of classes	No. of examples	References
Pima-Indians-diabetes	8	2	768	[51]
Orlraws10P	10304	10	100	[52]
Dermatology	34	6	366/**358**	[53]
Adult	14	2	48,842/**45,222**	[54]
Gisette	5000/**4891**	23	7000	[55]
Humanactivity	60/**57**	5	24,075	[56, 57]
Coil100	1024	100	7200	[58]
Gli_85	22,283/**22,259**	2	85	[59]
Orl_32×32	1024/**1023**	40	400	[60]
WarpAR10P	2420/**2251**	10	210	[61]
Yale_32×32	1024	15	165	[62]

Values in bold refer to the number of features, classes, and examples that remained in the datasets after the preprocessing stage

Consequently, this choice mitigates the potential impact of variations that may occur in each training instance on the outcomes produced by the feature selection procedures. Another advantage of the LDA classifier is that it does not require a numerical procedure to estimate the model parameters. As a result, the estimation process of the LDA model is significantly faster compared to classifiers, whose parameters are estimated under the training process. The parameters of each LDA classifier were estimated according to the 10-fold cross-validation procedure on 80% of data chosen randomly from the dataset. The remaining 20% of data was used to test the generalisation capabilities of the final classification model returned by the algorithm. The LDA classifier was employed in all algorithms tested in the paper.

In the case of GAAM and Melting GAAM, the three parameters shared by both algorithms (*M*, *N*, and *T*) were set at the same levels as in GAAMmf. In addition, for Melting GAAM, an extra parameter needed to be set: the accuracy threshold. This parameter informs the algorithm that the current number of genes has achieved satisfactory accuracy, and the algorithm should proceed with $N = N-1$ genes. We assumed that we would be satisfied with the classifier of 90% accuracy, and hence we set the accuracy threshold at 90%. Unfortunately, setting the accuracy threshold beforehand can be challenging as it depends on the characteristics of the dataset. As discussed in the Sect. 4, our threshold was too high for some datasets, resulting in no feature reduction, and too low for others, leading to the convergence of the algorithm to individuals with low classification accuracy.

For Holland GA, the classic scheme was employed, utilizing the two most popular genetic operations, flip mutation (with a probability of *0.1*) and one-point crossover (with a probability of *1*). The selection process was performed with the tournament method (the *tourN* parameter was set to *2*). The two primary algorithm parameters, the number of individuals in the mother population and the number of iterations, were set to the same levels as in GAAMmf. The fitness function was composed of accuracy and penalty terms, where the penalty term was introduced to penalise individuals for having too many genes. Both terms were assigned equal importance (6).

$$fitness(i) = 0.5 * acc(i) + 0.5 * \frac{P - features(i)}{P}, \tag{6}$$

where *fitness(i)*—the fitness of individual *i*, *acc(i)*—accuracy of the classifier equipped with features encoded in individual *i*, *P*—number of all features in the feature set, and *features(i)*—number of features encoded in individual *i*.

The primary parameters of NSGA-II, the last GA employed in the experiments (number of individuals in the population, mutation and crossover probability, and number of iterations), were set to the same levels as those for the Holland algorithm. Binary coding of features was applied, and the algorithm scheme proposed in [43] was implemented.

In addition to the four GAs, the set of reference methods employed in the experiments also included four non-genetic feature selection techniques: one filter (CFS), one embedded method (Lasso), and two wrappers (SFS and IniPG). To ensure comparability with the genetic algorithms, the upper boundary of the feature set size was set for all four methods at the same level as that for all GAs (8 for

Pima-Indians-diabetes, 14 for *Adult*, and 20 for the remaining datasets). All parameters required to run IniPG algorithm were set at the levels reported in [11].

The experiments were conducted on a machine with the following specification: Processor AMD Ryzen 5 1400 Quad-Core Processor CPU @ 3.20 GHz, 16GB RAM, Windows 10 Pro x64.

4 Results and discussion

In order to showcase the features of the proposed algorithm, we conducted a series of experiments. The first experiment aimed to demonstrate the impact of the two parameters incorporated in the algorithm for controlling the significance of the two competing criteria, namely classification accuracy (*accWeight*) and the number of features (*fsWeight*). Subsequently, we compared GAAMmf with four other genetic algorithms (GAs) capable of performing feature selection, namely GAAM, Melting GAAM, Holland with a penalty term, and NSGA-II. To facilitate the visual presentation of the results obtained from the first two experiments, both were executed on a single dataset only (*Gisette* in the first experiment; *Humanactivity* in the second experiment). Finally, we compared the performance of GAAMmf with all eight reference algorithms across the eleven datasets described in Sect. 3.

4.1 GAAMmf parameters (Gisette dataset)

This subsection provides an overview of GAAMmf's performance on the *Gisette* dataset, described in Sect. 3. To showcase the impact of the *accWeight* and *fsWeight* parameters on the algorithm's results, we executed the algorithm three times, each time with different values of both parameters. For the initial run, we set both parameters to 1 ($accWeight = 1$, $fsWeight = 1$). For the second and third runs, we doubled the significance of one of the fitness criteria, the number of features criterion ($accWeight = 1$, $fsWeight = 2$) in run 2 and the accuracy criterion ($accWeight = 2$, $fsWeight = 1$) in run 3. Each run was conducted over a period of 100 iterations. Figure 1 illustrates the average validation accuracy and the number of features encoded in the best individual returned by the algorithm for each iteration of each run. Additionally, Table 2 shows the average accuracy of the best individual found for different numbers of features.

As presented in Table 2, the algorithm produced comparable results across all three levels of *accWeight* and *fsWeight*. The highest classification accuracy was equal to 93.15% for an equal value of both parameters, 92.04% for the doubled significance of the number-of-features criterion ($fsWeight = 2$), and 94.03% for the doubled importance of the accuracy criterion ($accWeight = 2$). Regarding the second criterion, the algorithm attained the smallest number of features (i.e., 10) with a doubled *fsWeight* parameter. The other algorithm runs returned feature sets composed of 15 features. Upon comparing the three sets of results, the algorithm obtained the most promising outcomes with a doubled accuracy weight. As demonstrated in the last two columns of Table 2, by applying greater pressure on the accuracy criterion,

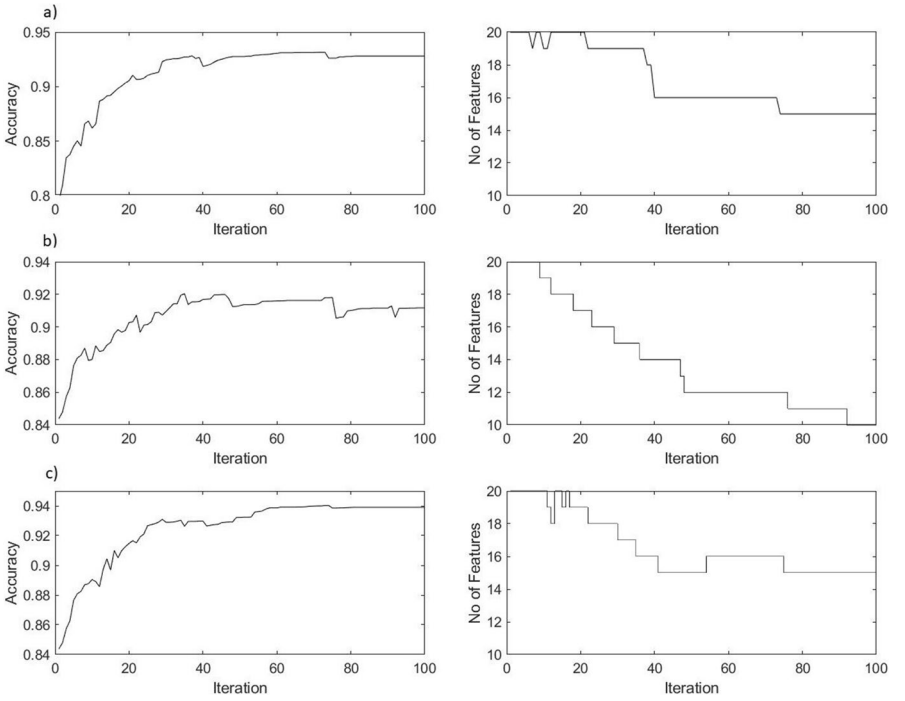

Fig. 1 The GAAMmf performance for the *Gisette* dataset; the plots on the left present the average validation accuracy of the best individual returned in each iteration, the plots on the right present the number of features encoded in that individual; the rows of plots present results for different levels of *accWeight* and *fsWeight* parameters: **a** $accWeight = 1, fsWeight = 1$; **b** $accWeight = 1, fsWeight = 2$; **c** $accWeight = 2, fsWeight = 1$

Table 2 The highest accuracy obtained for different numbers of features for the *Gisette* dataset

accWeight = 1, fsWeight = 1		accWeight = 1, fsWeight = 2		accWeight = 2, fsWeight = 1	
Acc (%)	No. of features	Acc (%)	No. of features	Acc (%)	No. of features
92.81 (90.57)	15	91.17 (90.07)	10	93.91 (92.64)	15
93.15 (90.86)	16	91.29 (90.00)	11	**94.03** (92.61)	16
91.81 (90.36)	17	91.81 (90.36)	12	93.04 (91.93)	17
92.68 (91.57)	18	91.73 (90.36)	13	93.10 (91.50)	18
92.04 (91.93)	19	91.99 (90.64)	14	92.83 (90.36)	19
91.05 (90.07)	20	**92.04** (90.71)	15	91.81 (89.07)	20
		90.90 (89.29)	16		
		90.72 (89.57)	17		
		89.83 (87.79)	18		
		88.84 (87.50)	19		
		88.70 (87.79)	20		

The values in brackets present the accuracy calculated over the test set. The results come from three runs of GAAMmf performed with different levels of *accWeight* and *fsWeight* parameters

we forced the algorithm to conduct a more thorough search amongst the individuals with the same number of features. Consequently, the algorithm returned individuals of greater accuracy for each number of features in comparison to the other two cases. Since we were interested in high accuracy in the two following experiments, we utilized the variant with $accWeight = 2$ and $fsWeight = 1$ in both.

As illustrated in Fig. 1, the performance of the algorithm was consistent across all levels of the parameters *accWeight* and *fsWeight*, leading to individuals with high accuracy and a small number of features. The convergence rate varied across runs and was dependent on the parameter levels, with higher values of *fsWeight* resulting in a more rapid reduction of features but with some fluctuations in accuracy (Fig. 1b). Conversely, for higher values of *accWeight* (Fig. 1c), the algorithm demonstrated a highly stable accuracy performance, albeit with a slower rate of feature reduction.

4.2 Comparison of GAAMmf with genetic reference methods (Humanactivity dataset)

The second experiment aimed to compare the performance of GAAMmf with four other GAs (GAAM, Melting GAAM, Holland with a penalty term, and NSGA-II) on the *Humanactivity* dataset. Results are presented in Tables 3, 4 and Fig. 2. Table 3 shows the average validation accuracy of the best individuals identified for various numbers of features, Table 4 presents the processing time required to complete 100 iterations, and Fig. 2 compares the performance of all five algorithms across 100 iterations.

Table 3 demonstrates that the highest accuracy obtained by all four reference algorithms was similar, approximately 96–97%. The best accuracy of 97.80% was obtained with GAAM, followed by Melting GAAM with 97.30%, NSGA-II with 97.22%, and finally Holland with 96.98%. The accuracy of the best individual provided by the GAAMmf algorithm was slightly lower (96.99%) than the accuracy of the reference algorithms, but the difference was tiny, ranging from 0.23 to 0.81%.

Although the accuracy obtained with all five algorithms was similar, the number of features in individuals with the highest accuracy varied significantly. For the algorithms using a binary coding scheme (NSGA-II and Holland), the feature reduction was relatively small, with the best individuals containing 22 features (NSGA-II). In the case of GAAM, there was no feature reduction at all; the change from an initial 20 to 19 genes was caused only by a duplicated feature in the individual. The highest reduction was achieved for GAAMmf and Melting GAAM. In the case of GAAMmf, an individual with only 7 features had an accuracy (96.54%) of only 1.26% lower than the best individual in the table (20 features, 97.80%).

When comparing the feature reduction plots (the plots on the right-hand side of Fig. 2), it can be observed that the reduction process for GAAMmf and Melting GAAM remained relatively stable over time, gradually decreasing until the final number of features was reached. In contrast, the number of features in individuals produced by the two other algorithms designed for feature reduction (Holland and NSGA-II) slightly fluctuated during the initial period. Since the

Table 3 The highest accuracy returned by five GAs for different numbers of features (*Humanactivity* dataset)

GAAMmf		GAAM		Melting		NSGA-II		Holland	
Acc (%)	F	Acc (%)	F	Acc (%)	F	Acc (%)	F	Acc (%)	F
96.54 (96.03)	7	97.65 (97.51)	19	84.35 (84.67)	1	96.55 (96.43)	22	**96.98** (96.95)	27
96.58 (96.12)	8	**97.80** (97.59)	20	94.85 (94.50)	2	96.99 (97.26)	24	96.95 (96.93)	28
96.62 (96.24)	9			95.36 (94.56)	3	96.98 (97.23)	25	96.93 (97.03)	32
96.76 (96.24)	10			95.52 (94.75)	4	96.88 (96.68)	26	96.66 (96.66)	33
96.81 (96.28)	11			95.89 (95.06)	5	97.09 (96.93)	27		
96.62 (96.22)	12			96.28 (95.78)	6	96.61 (96.81)	28		
96.93 (96.78)	13			96.49 (95.93)	7	**97.22** (97.05)	29		
96.98 (96.78)	14			96.66 (96.37)	8				
96.99 (96.68)	15			96.76 (96.32)	9				
96.95 (96.74)	16			96.90 (96.51)	10				
95.40 (95.04)	17			96.90 (96.39)	11				
				96.90 (96.41)	12				
				96.98 (96.39)	13				
				97.30 (96.49)	14				
				96.88 (96.45)	15				
				96.91 (96.51)	16				
				96.71 (96.59)	17				
				96.63 (96.55)	18				

Acc average classification accuracy, *F* number of features

The values in brackets present the accuracy calculated over the test set

goal of the last algorithm (GAAM) was not feature reduction, the number of features in individuals produced by this algorithm remained largely consistent across all 100 iterations.

Finally, regarding the processing time, it can be observed (Table 4) that GAAMmf required a significant amount of time to complete the required number of iterations (1 h 57 min), especially when compared to Holland (18 min), Melting GAAM (24 min) and NSGA-II (1 h 02 min). One of the reasons for such a long processing time was a significantly higher number of individuals evaluated by GAAMmf in each iteration. While Holland and NSGA-II evaluated only 10 individuals per iteration, i.e., 1000 individuals in 100 iterations, GAAMmf evaluated between 94 and 190 individuals in each iteration (10 mother individuals, 10 crossed-over, and from 70 to 170 mutated, depending on the average number of features in individuals in the current population). Although Melting GAAM started with the same number of individuals as GAAMmf, it quickly reduced the number of individuals needing evaluation to only one per iteration. Nevertheless, regardless of the reason, the long processing time should be considered a limitation of the proposed algorithm.

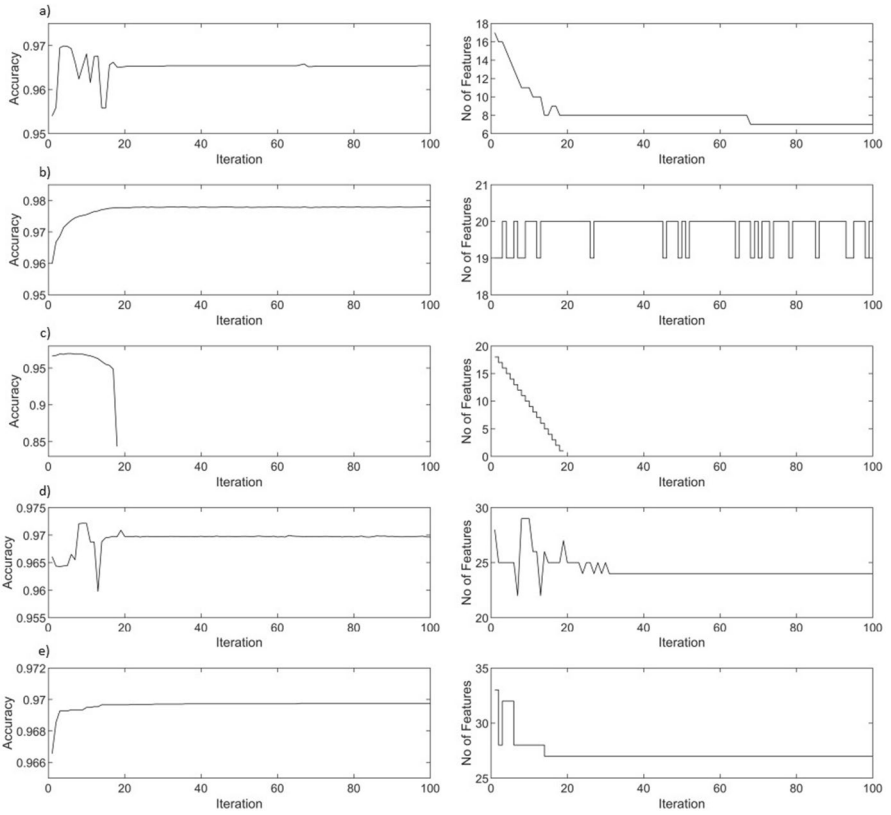

Fig. 2 The algorithms' performance for the *Humanactivity* dataset; the plots on the left present the average validation accuracy of the best individual returned in each iteration, the plots on the right present the number of features encoded in that individual; the rows of plots present results for different algorithms: **a** GAAMmf; **b** GAAM; **c** Melting GAAM; **d** NSGA-II; **e** Holland

Table 4 The processing time needed to complete 100 iterations by each GA (results for *Humanactivity* dataset)

GAAMmf	GAAM	Melting	NSGA-II	Holland
1 h 57 min	5 h 14 min	24 min	1 h 02 min	18 min

4.3 Comparison of GAAMmf with reference methods (all datasets)

In the previous subsection, we demonstrated that the individuals returned by GAAMmf for the *Humanactivity* dataset had slightly lower accuracy but were composed of a significantly smaller number of features compared to those produced by most other GAs (with the exception of Melting GAAM). The objective of the experiment reported in this section was to determine whether this observation is consistent

across datasets of different numbers of features, classes, and examples. Unlike in the two previous sections, we do not present here the detailed results obtained in individual iterations. Instead, for each method and dataset, we report the characteristics of the feature set with the best classification accuracy. The comparison of results achieved by GAAMmf and eight reference methods (GAAM, Melting GAAM, Holland, NSGA-II, CFS, Lasso, SFS, IniPG) for eleven datasets described in Sect. 3 (Table 1) is presented in Table 5.

To test the statistical significance of GAAMmf's results against those obtained with the reference methods, we conducted a series of one-sample tests with a significance level (α) set to 0.05. Each test tested the hypothesis H0: *the difference between the accuracy (or number of features) of individuals returned by GAAMmf and one of the reference methods is equal to zero* against the alternative hypothesis H1: *the difference between the accuracy (or number of features) of individuals returned by GAAMmf and one of the reference methods is not equal to zero*.

To verify this set of hypotheses, we first calculated the differences between the results obtained by GAAMmf and each of the reference methods. This process yielded a set of 12 samples, consisting of 6 samples for differences in accuracy and 6 samples for differences in the number of features. Subsequently, we assessed the normality condition for each sample using the one-sample Lilliefors test with α set to 0.05. Finally, since not all samples met the normality condition, we applied a one-sample Wilcoxon signed-rank test to test the differences significance. The results of the Wilcoxon test are presented in Fig. 3, which shows the mean differences in accuracy (Fig. 3a) and the number of features (Fig. 3b) calculated between GAAMmf and each of the reference methods (the actual *p-value* is provided for all statistically significant results).

As shown in Table 5, not all methods produced results for all datasets. The Holland and NSGA-II algorithms encountered problems in the classifiers' training process for the individuals from seven datasets (*Orlraws10P*, *Gisette*, *Coil100*, *Gli_85*, *Orl_32×32*, *WarpAR10P*, and *Yale_32×32*). For all of those datasets, due to the unfavourable ratio of the number of features to the number of examples, the classification process could not be completed because the covariance matrices did not meet the positive definiteness condition. Since only four valid results were possible to obtain for the Holland and NSGA-II algorithms, they were excluded from the statistical tests. At first, a similar problem was encountered with IniPG. However, we managed to overcome it by slightly changing the parameters proposed in [11]. Our modification was to initialize all the particles with a small number of features (the same as was used for other algorithms) instead of using particles with sparse and dense initialization.

When analysing the classification accuracy of the algorithms presented in Table 5, GAAM outperformed the other methods for almost all datasets. Only for three datasets, *Adult*, *Gisette*, and *Orl_32×32*, other algorithms returned classifiers with marginally higher accuracy. The second place was shared between GAAMmf (eight datasets), Holland (two datasets), and CFS (one dataset). Comparing the accuracy differences averaged over eleven datasets (Fig. 3a), it can be noticed that GAAMmf exhibited superior performance compared to the five reference methods: Melting GAAM, Lasso, CFS, IniPG, and SFS. In the case of all of those methods,

Table 5 The algorithms' results across all datasets

	GAAMmf	GAAM	Melting	NSGA-II	Holland	Lasso	CFS	IniPG	SFS
Classification accuracy (validation phase) [%]									
Pima-Indians-diabetes	78.12	**78.28**	77.36	77.87	72.77	77.19	73.13	77.36	77.17
Orlraws10P	98.75	**100.00**	89.79	–	–	80.00	57.50	97.50	96.26
Dermatology	97.91	**98.79**	89.19	97.90	98.35	96.15	88.10	94.97	96.17
Adult	79.04	78.78	78.88	**79.53**	79.06	78.73	78.75	77.92	77.88
Gisette	93.91	92.31	89.97	–	–	91.20	89.02	83.15	**94.04**
Humanactivity	96.54	**97.65**	84.35	96.55	96.98	90.76	95.21	93.59	93.25
Coil100	73.67	**74.02**	73.18	–	–	51.67	40.47	68.53	71.23
Gli_85	94.29	**100.00**	79.06	–	–	92.38	95.48	93.45	94.20
Orl_32×32	95.37	95.35	89.86	–	–	66.56	55.31	**96.25**	90.94
WarpAR10P	95.57	**99.55**	88.66	–	–	78.82	40.64	94.51	92.36
Yale_32×32	92.17	**94.25**	89.48	–	–	52.2	35.55	84.2	75.16
Mean	*90.48*	*91.73*	*84.53*	–	–	*77.79*	*68.11*	*87.40*	*87.15*
	GAAMmf	GAAM	Melting	NSGA-II	Holland	Lasso	CFS	IniPG	SFS
Classification accuracy (test phase) [%]									
Pima-Indians-diabetes	78.57	75.32	75.97	72.73	78.57	74.68	79.87	77.27	77.27
Orlraws10P	95.00	**85.00**	**75.00**	–	–	**70.00**	70.00	**85.00**	**85.00**
Dermatology	97.22	93.06	87.50	91.67	93.06	90.28	83.33	94.44	91.67
Adult	79.14	78.89	78.53	79.39	79.13	78.53	78.88	77.26	77.67
Gisette	92.64	90.57	87.21	–	–	90.29	88.86	83.79	92.57
Humanactivity	96.03	97.51	84.67	96.43	96.95	90.80	95.18	93.31	93.31
Coil100	72.50	71.60	69.93	–	–	49.65	41.04	67.87	68.82
Gli_85	**82.35**	**76.47**	**58.82**	–	–	**82.35**	88.24	**76.47**	**76.47**
Orl_32×32	91.25	83.75	77.50	–	–	67.50	52.50	96.05	80.00
WarpAR10P	**84.62**	**84.62**	**61.54**	–	–	**65.38**	61.54	88.46	**69.23**
Yale_32×32	**75.70**	**69.70**	**75.76**	–	–	**36.36**	**33.33**	**66.67**	**57.58**
Mean	*85.91*	*82.41*	*75.68*	–	–	*72.35*	*70.25*	*82.42*	*79.05*
	GAAMmf	GAAM	Melting	NSGA-II	Holland	Lasso	CFS	IniPG	SFS
Number of features									
Pima-Indians-diabetes	5	5	7	5	**2**	8	8	5	5
Orlraws10P	6	20	**3**	–	–	20	20	39	7
Dermatology	8	19	**4**	17	11	20	20	19	9
Adult	3	13	14	7	3	14	14	**2**	3
Gisette	15	20	**8**	–	–	18	20	20	20
Humanactivity	7	19	**1**	22	27	20	20	8	14
Coil100	20	20	20	–	–	20	20	20	20
Gli_85	2	19	**1**	–	–	18	20	40	2
Orl_32×32	16	20	**11**	–	–	19	20	78	16
WarpAR10P	13	19	**7**	–	–	20	20	40	9

Table 5 (continued)

	GAAMmf	GAAM	Melting	NSGA-II	Holland	Lasso	CFS	IniPG	SFS
Yale_32×32	15	19	**12**	–	–	20	20	59	15
Mean	*10*	*18*	*8*	–	–	*18*	*18*	*30*	*11*
Dataset	GAAMmf	GAAM	Melting	NSGA-II	Holland	Lasso	CFS	IniPG	SFS
Processing time									
Pima-Indians-diabetes	18′	28′	24′	44′	13′	2″	1″	12′	3″
Orlraws10P	17′	15′	9′	–	–	15″	2″	10′	23′
Dermatology	24′	16′	9′	1 h 19′	9′	2″	1″	11′	10″
Adult	5 h 18′	2 h 56′	4 h 23′	9 h 10′	2 h 37′	13″	1″	42′	10″
Gisette	6 h 54′	4 h 11′	41′	–	–	2′	6″	3 h 47″	2 h 38′
Humanactivity	8 h 47′	5 h 31′	24′	49 h 12′	14 h 11′	15″	1″	4 h 3′	5′
Coil100	68 h 55′	19 h 32′	65 h 17′	–	–	22″	1″	5 h 56′	4 h 11′
Gli_85	57′	1 h 15′	9′	–	–	30″	8″	7′	29′
Orl_32×32	3 h 17′	3 h 53′	1 h 49′	–	–	5″	1″	21′	16′
WarpAR10P	1 h 21′	1 h 37′	29′	–	–	6″	1″	9′	10′
Yale_32×32	1 h 28′	1 h 57′	9′	–	–	3″	1″	12′	7′
Mean	*8 h 54′*	*3 h 48′*	*6 h 43′*	–	–	*0.01′*	*0.01′*	*1 h 21′*	*45′*

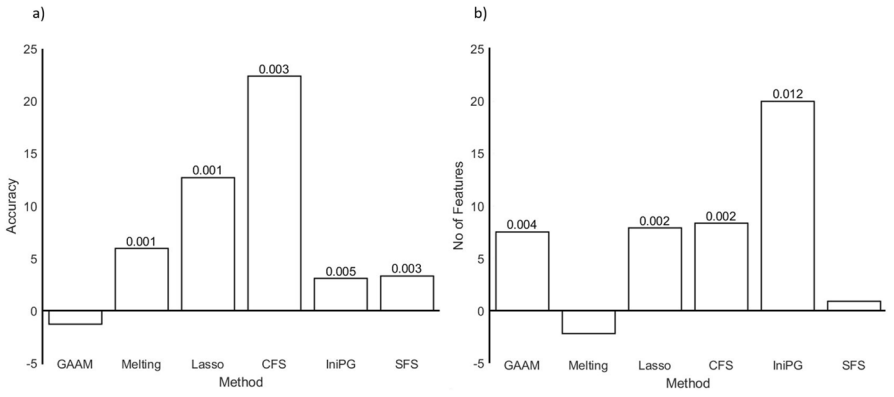

Fig. 3 The statistical significance of differences calculated between GAAMmf and other methods in terms of **a** classification accuracy, **b** number of features; *p value* for each significant difference is presented over the corresponding bar

the difference in accuracy was statistically significant. In fact, the only algorithm that performed better than GAAMmf in terms of accuracy was the original GAAM. However, the difference in accuracy between those two algorithms was insignificant.

As shown in Table 5, although Holland and NSGA-II returned individuals with accuracies comparable to those generated by GAAM-based algorithms, their application can sometimes be challenging. This is due to the fact that both algorithms employ a binary coding scheme, which begins the search process with roughly half

of the total number of features. As a result, their individuals can be difficult to evaluate in terms of classification accuracy when applied to datasets with an unfavorable ratio between the number of features and the number of examples (e.g., *Gisette* and *Orlaws10P*). In contrast, all three GAAM-based algorithms permit the initial selection of the number of genes (i.e., features) in individuals. Consequently, they are not affected by the dimensionality 'curse' and can be applied to datasets with arbitrary characteristics.

Moreover, it is worth noting that some non-genetic feature selection methods exhibited significantly poorer performance when confronted with multiclass problems. For example, as can be noticed in Table 5, Lasso and CFS performed much worse in the case of most multiclass datasets (apart from *Dermatology* and *Gisette* (analysed by Lasso)). The most extreme drop in accuracy measured between GAAMmf and the aforementioned methods could be observed for *Orlraws10P*, *Coil100*, *Orl_32×32*, *WarpAR10P*, and *Yale_32×32*.

The second section of Table 5 displays the accuracy of the final classification model estimated for each algorithm using the 20% of data that was not utilized in the parameters estimation process. We marked with bold font all the cases where the test accuracy was 10% lower than the corresponding validation accuracy reported in the first part of the table. Analysis of the table reveals that the parameters of most classifiers were correctly estimated, with the test accuracy being only slightly lower or, in some cases, slightly higher than the validation accuracy. Additionally, it is evident that the occurrence of overfitting behaviour primarily depended on the characteristics of the datasets rather than the algorithms themselves. Notably, for four datasets (Orlaws10P, Gli_85, WarpAR10P, and Yale_32×32), almost all classifiers exhibited overfitting behaviour, while for the remaining datasets, overfitting was not observed.

Regarding the average accuracy calculated across all eleven datasets, the classic GAAM outperformed other algorithms in terms of average validation accuracy, achieving a score of 91.73%. However, it displayed the poorest generalization capabilities among the estimated classifiers. On the other hand, GAAMmf demonstrated slightly lower average validation accuracy (90.48%) compared to GAAM, but it emerged as the winner in terms of test accuracy and generalization capabilities.

Concerning the third section of Table 5, which presents the number of features encoded in individuals with the highest accuracy, the outcomes significantly differ from those presented in the first section of the table. Here, Melting GAAM returned individuals with the fewest features, averaging at 8 features. GAAMmf followed closely with 10 features, while SFS obtained the third-best performance with 11 features. Other algorithms yielded significantly larger feature sets (Fig. 3b).

The comparison across different datasets presents one undesirable feature of Melting GAAM that motivated us to seek an approach to balance the accuracy and number of feature criteria in the feature selection process. By fixing the accuracy, the algorithm might halt the search process with individuals that are far from optimal, either in terms of accuracy or the number of features. When the accuracy threshold is underestimated, the algorithm terminates with individuals of a much worse accuracy than optimal (for *Orlraws10P*, the accuracy of Melting GAAM was over 10% worse than that of GAAM). Conversely, when the accuracy threshold exceeds the accuracy that can be achieved in the given dataset, the algorithm focuses

entirely on the accuracy criterion, and no feature reduction is achieved. This situation was observed for *Adult* dataset, where Melting GAAM stopped with 14 features, although the individual composed of only three features returned by GAAMmf provided even better accuracy.

In the final part of Table 5, the total processing time required by all algorithms to complete the task was compared. The ranking of the methods in this section of the table was consistent with expectations. The quickest method was CFS, followed by Lasso. All wrappers required significantly longer time to fulfil the task. Among the wrappers, the SFS method was at the forefront, followed by IniPG, GAAM, Melting GAAM, and GAAMmf. The two least time-efficient methods were Holland and NSGA-II, respectively.

5 Conclusions

This study developed and evaluated a genetic algorithm for feature selection (GAAMmf), which is a modified version of a genetic algorithm with aggressive mutation. The new version of the algorithm was developed to overcome the limitations of both preceding GAAM-based algorithms, the original GAAM and Melting GAAM. The original GAAM is focused on feature subsets of a fixed size. Hence, it optimises the feature set in terms of classification accuracy but not the number of features. Conversely, Melting GAAM fixes the classification accuracy and optimises only the size of the feature space. GAAMmf combines both criteria and, at the same time, optimises the classification accuracy and the size of the feature set, similar to Holland with a penalty term and NSGA-II. One distinguishing feature of GAAMmf is its ability to start the feature reduction process from either the entire feature set or an arbitrarily chosen number of features, which is impossible with the direct binary coding scheme. This feature makes GAAMmf applicable to datasets of any characteristic.

In summary, the main benefit of GAAMmf is its ability to be run without tedious tuning of parameters. All parameters can be set to the levels used in the experiments described in the paper, which may not be optimal in terms of processing time but will produce a sufficiently small feature set of satisfactory classification accuracy for most datasets. On the other hand, the main limitation of GAAMmf is its processing time. The aggressive mutation used in the evolution process allows the algorithm to explore different subspaces of the search space, but also produces a large number of new individuals that must be evaluated. Attempts were made to overcome this problem by introducing the concept of mutation probability, but the processing time is still unsatisfactory. Hence, in future work, we plan to apply the concept of keeping track of previously evaluated individuals to avoid the cost of their reevaluation.

Author contributions Conceptualization: IR; Literature review: KL; Methodology: IR, KL; Formal analysis IR; Investigation: KL; Writing—original draft preparation: IR, KL; Software preparation: KL; Writing—review and editing: IR; Supervision: IR.

Funding This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Declarations

Conflict of 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.

Consent for of publication The authors declare that the results/data/figures in this manuscript have not been published elsewhere, nor are they under consideration (from you or one of your Contributing Authors) by another publisher.

References

1. H.N. Bach, X. Bing, Z. Mengjie, A survey on swarm intelligence approaches to feature selection in data mining. Swarm Evol. Comput. **54**, 100663 (2020)
2. I. Kononenko, Estimating Attributes: Analysis and Extensions of Relief, in *Machine Learning: ECML-94*. ed. by L. De Raedt, F. Bergadano (Springer Verlag, Berlin, 1994), pp.171–182
3. M.A. Hall, Correlation-based feature selection for machine learning. The University of Waikato (1999)
4. J. Kittler, Feature set search algorithms. Pattern Recognit Signal Process 41–60 (1978)
5. I. Rejer, EEG feature selection for BCI based on motor imaginary task. Found. Comput. Decis. Sci. **37**(4), 283–292 (2012)
6. R. Burduk, Recognition Task with Feature Selection and Weighted Majority Voting Based on Interval-Valued Fuzzy Sets in Computational Collective Intelligence, in *Technologies and Applications, Lecture Notes in Computer Science*. ed. by N.T. Nguyen, K. Hoang, P. Jedrzejowicz (Springer, Berlin, 2012)
7. I. Guyon, J. Weston, S. Barnhill, V. Vapnik, Gene selection for cancer classification using support vector machines. Machine Learning **46**(1–3), 389–422 (2002)
8. R. Tibshirani, Regression shrinkage and selection via the lasso. J. R. Stat. Soc. Ser. B (Methodological) **58**(1), 267–288 (1996)
9. M. Sharma, P. Kaur, A comprehensive analysis of nature-inspired meta-heuristic techniques for feature selection problem. Arch. Comput. Methods Eng. (2020)
10. B. Tran, B. Xue, M. Zhang, S. Nguyen, Investigation on particle swarm optimisation for feature selection on high-dimensional data: local search and selection bias. Connect. Sci. **28**, 270–294 (2016)
11. B. Xue, M. Zhang, W. Browne, Novel Initialisation and Updating Mechanisms in PSO for Feature Selection in Classification, in *Applications of Evolutionary Computation. Lecture Notes in Computer Science*. (Springer, Berlin, 2013), pp.428–438
12. R. Poli, J. Kennedy, T. Blackwell, Particle swarm optimization (PSO). A tutorial. Swarm Intell. **1**, 33–57 (2007)
13. F. Marini, B. Walczak, Particle swarm optimization (PSO). A tutorial. Chemom. Intell. Lab. Syst. **149**, 153–165 (2015)
14. D. Wang, D. Tan, L. Liu, Particle swarm optimization. Soft. Comput. **22**, 387–408 (2018)

15. M. Dorigo, M. Birattari, T. Stutzle, Ant colony optimization. IEEE Comput. Intell. Mag. **1**(4), 28–39 (2006)
16. S.G. Yaseen, N.M.A. Al-Slamy, Ant colony optimization. IJCSNS Int. J. Comput. Sci. Netw. Secur. **8**(6), 351 (2008)
17. H.R. Kanan, K. Faez, S.M. Taheri, Feature selection using ant colony optimization (ACO): a new method and comparative study in the application of face recognition system. Adv. Data Min. Theor. Asp. Appl. **4597**(6), 63–76 (2007)
18. A.L. Bolaji, A.T. Khader, M.A. Al-Betar, M.A. Awadallah, Artificial bee colony algorithm, its variants and applications: a survey. J. Theor. Appl. Inf. Technol. **47**(2), 434–459 (2013)
19. D. Karaboga, B. Basturk, Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems. Found. Fuzzy Logic Soft Comput. **4529**, 789–798 (2007)
20. W.-F. Gao, S.-Y. Liuy, A modified artificial bee colony algorithm. Comput. Oper. Res. **39**, 687–697 (2012)
21. A. Madasu, S. Elango, Efficient feature selection techniques for sentiment analysis. Multimed. Tools Appl. **79**, 6313–6335 (2020)
22. M. Devaney, A. Ram, Efficient feature selection in conceptual clustering, in Machine Learning: Proceedings of the Fourteenth International Conference, vol. 79 (1997), pp. 6313–6335
23. H. Shi, H. Li, D. Zhang, C. Cheng, X. Cao, An efficient feature generation approach based on deep learning and feature selection techniques for traffic classification. Comput. Netw. **132**, 81–98 (2018)
24. I. Rejer, Genetic algorithms for feature selection for brain computer interface. J. Artif. Intell. Res. **1**, 1–15 (2015)
25. I. Rejer, K. Lorenz, Feature selection with NSGA and GAAM in EEG signals domain, in 8th International Conference on Human System Interaction (HSI) (2015), pp. 94–98
26. K. Lorenz, Przeglad algorytmow genetycznych stosowanych w procesie selekcji cech wyekstrahowanych z sygnału eeg. Młodzi naukowcy dla polskiej nauki (2013)
27. D.A. Peterson, J.N. Knight, M.J. Kirby, C.W. Anderson, M.H. Thaut, Feature selection and blind source separation in an EEG-based brain-computer interface. EURASIP J. Adv. Signal Process. **2005**(19), 1–13 (2005)
28. M. Sewell, Feature selection. https://pdfs.semanticscholar.org/aacd/187f333a60718387d4f42a18929d18203e0e.pdf. Accessed 22 May 2022
29. M. Kołodziej, A. Majakowski, J.R. Rak, A new method of EEG classification for BCI with feature extraction based on higher order statistics of wavelet components and selection with genetic algorithms, in ICANNGA (2011), pp. 280–289
30. E.B. Garrett, D. Baum, C. Boneh, Where genetic algorithms excel. Evol. Comput. **9**(1), 93–124 (2001)
31. D. Garrett, D.A. Peterson, C. Anderson, M.H. Thaut, Comparison of linear, nonlinear, and feature selection methods for EEG signal classification. IEEE Trans. Neural Syst. Rehabil. Eng. **11**(2), 141–145 (2003)
32. H. Lakany, B.A. Conway, Understanding intention of movement from electroencephalograms. Expert Syst. **24**(5), 295–304 (2007)
33. D. Whitley, J.R. Beveridge, C. Guerra-Salcedo, C. Graves, Messy genetic algorithms for subset feature selection, in Proceedings of the 7th International Conference on Genetic Algorithms (1997)
34. J. Yang, V. Honavar, Feature subset selection using genetic algorithm, in feature extraction, construction and selection. Springer Int. Ser. Eng. Comput. Sci. **453**, 117–136 (1998)
35. I. Koprinska, Feature selection for brain-computer interfaces. J. Artif. Intell. Res. 100–111 (2010)
36. M. Bereta, P. Jarosz, Algorytmy genetyczne. https://michalbereta.pl/dydaktyka/ae/lab_genetyczne/laboratorium_genetyczne.pdf. Accessed 22 May 2022
37. Z. Mingyuan, F. Chong, J. Luping, K. Tang, M. Zhou, Feature selection and parameter optimization for support vector machines: a new approach based on genetic algorithm with feature chromosomes. Expert Syst. Appl. **38**, 5197–5204 (2011)
38. R. Leardi, Application of genetic algorithm-pls for feature selection in spectral data sets. J. Chemometrics **1**, 643–655 (2000)
39. I.P. Benitez, A.M. Sison, R.P. Medina, An improved genetic algorithm for feature selection in the classification of disaster-related twitter messages, in 2018 IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE). IEEE (2018), pp. 238–243
40. I. Rejer, K. Lorenz, Classic genetic algorithm vs. genetic algorithm with aggressive mutation for feature selection for brain-computer interface. Prz. Elektrotech. **91**(2), 98–102 (2015)

41. B. Huang, B. Buckley, T.M. Kechadi, Multi-objective feature selection by using NSGA-II for customer churn prediction in telecommunications. Expert Syst. Appl. **37**(5), 3638–3646 (2010)
42. U. Tekguc, H. Soyel, H. Demirel, Feature selection for person independent 3d facial expression recognition using NSGA-II, in 24th International Symposium on Computer and Information Sciences, vol. 1, (2009), pp. 35–38
43. K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)
44. T.M. Hamdani, J.M. Won, A.M. Alimi, F. Karray, Multi-objective feature selection with NSGA-II. Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (2007), pp. 240–247
45. H. Soyel, U. Tekguc, H. Demirel, Application of NSGA-II to feature selection for facial expression recognition. Comput. Electr. Eng. **37**(6), 1232–1240 (2011)
46. I. Rejer, K. Lorenz, Feature selection with NSGA and GAAM in EEG signals domain, in 8th International Conference on Human System Interaction (HSI) (2015)
47. I. Rejer, M. Twardochleb, Gamers involvement detection from EEG data with cGAAM—a method for feature selection for clustering. Expert Syst. Appl. **101**, 196–204 (2018)
48. M.J. Pontiveros, G.A. Solano, J.M. Diaz, J.D. Caro, Feature subset selection using genetic algorithm with aggressive mutation for classification problem, in TENCON 2021–2021 IEEE Region 10 Conference (TENCON) (2021)
49. Y.A. Baysal, S. Ketenci, I.H. Altas, T. Kayikcioglu, Multi-objective symbiotic organism search algorithm for optimal feature selection in brain computer interfaces. Expert Syst. Appl. **165**, 113907 (2021)
50. Rejer, I., Jankowski, J. fGAAM: a fast and resizable genetic algorithm with aggressive mutation for feature selection. Pattern Anal. Appl. (2021)
51. Pima-indians-diabetes dataset. https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database. Accessed 22 May 2022
52. Orlraws10P dataset. https://jundongl.github.io/scikit-feature/datasets.html. Accessed 22 May 2022
53. Dermatology dataset. https://archive.ics.uci.edu/ml/datasets/Dermatology. Accessed 22 May 2022
54. Adult dataset. https://archive.ics.uci.edu/ml/datasets/Adult. Accessed 22 May 2022
55. Gisette dataset. http://archive.ics.uci.edu/ml/datasets/Gisette. Accessed 22 May 2022
56. Humanactivity dataset (2019). https://www.mathworks.com/products/new_products/release2019b.html. Accessed 22 May 2022
57. Sensor HAR recognition App (2019), MATLAB Central File Exchange. https://www.mathworks.com/matlabcentral/fileexchange/54138-sensor-har-recognition-app. Accessed 22 May 2022
58. Coil100 dataset. https://www.cs.columbia.edu/CAVE/software/softlib/coil-100.php. Accessed 20 Feb 2023
59. GLI85 dataset. https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE4412. Accessed 20 Feb 2023
60. ORL dataset. http://www.cad.zju.edu.cn/home/dengcai/Data/FaceData.html. Accessed 20 Feb 2023
61. Dermatology dataset. https://jundongl.github.io/scikit-feature/datasets.html. Accessed 20 Feb 2023
62. Yale32x32 dataset. http://www.cad.zju.edu.cn/home/dengcai/Data/FaceData.html. Accessed 20 Feb 2023

Authors and Affiliations

Rejer Izabela[1] · Lorenz Krzysztof[1]

✉ Lorenz Krzysztof
klorenz@zut.edu.pl

Rejer Izabela
irejer@zut.edu.pl

[1] Faculty of Computer Science, West Pomeranian University of Technology in Szczecin, Żołnierska 49, Szczecin 71-210, Szczecin, Poland