symmetry

MDPI

Article

A New Feature Selection Method Based on a Self-Variant Genetic Algorithm Applied to Android Malware Detection

Le Wang [1], Yuelin Gao [2,*], Shanshan Gao [1] and Xin Yong [1]

[1] School of Computer Science and Engineering, Northern Minzu University, Yinchuan 750021, China; sdnywll@163.com (L.W.); 13008030953@163.com (S.G.); yongxin_Azure@163.com (X.Y.)
[2] Ningxia Province Key Laboratory of Intelligent Information and Data Processing, North Minzu University, Yinchuan 750021, China
* Correspondence: gaoyuelin@263.net; Tel.: +86-139-9510-0900

Abstract: In solving classification problems in the field of machine learning and pattern recognition, the pre-processing of data is particularly important. The processing of high-dimensional feature datasets increases the time and space complexity of computer processing and reduces the accuracy of classification models. Hence, the proposal of a good feature selection method is essential. This paper presents a new algorithm for solving feature selection, retaining the selection and mutation operators from traditional genetic algorithms. On the one hand, the global search capability of the algorithm is ensured by changing the population size, on the other hand, finding the optimal mutation probability for solving the feature selection problem based on different population sizes. During the iteration of the algorithm, the population size does not change, no matter how many transformations are made, and is the same as the initialized population size; this spatial invariance is physically defined as symmetry. The proposed method is compared with other algorithms and validated on different datasets. The experimental results show good performance of the algorithm, in addition to which we apply the algorithm to a practical Android software classification problem and the results also show the superiority of the algorithm.

Keywords: feature selection; machine learning; asexual; genetic algorithm; android malicious application detection

Citation: Wang, L.; Gao, Y.; Gao, S.; Yong, X. A New Feature Selection Method Based on a Self-Variant Genetic Algorithm Applied to Android Malware Detection. *Symmetry* **2021**, *13*, 1290. https://doi.org/10.3390/sym13071290

Academic Editors: Kóczy T. László and István A. Harmati

Received: 10 June 2021
Accepted: 14 July 2021
Published: 18 July 2021

1. Introduction

Data classification is one of the tasks of data mining in the field of machine learning and in the framework of pattern recognition [1]; the quality of the data has a significant impact on the performance of these data mining methods. When training machine learning models, irrelevant, redundant, and noisy data have an enormous impact on the time and the spatial complexity of the machine and can also affect the algorithm's performance. Therefore, pre-processing techniques for data are necessary [2]. In machine learning classification tasks, the dataset's size determines the number of features in the dataset, but not all features are helpful for training classifier models, and high-dimensional features can instead lead to dimensionality disasters. Data dimensionality reduction methods include feature extraction(FE), where features are transformed into a smaller dimension, and feature selection(FS) [3], where features are selected from the complete set of features to build a subset of features without transformation [4]. The method chosen in this paper is feature selection, the aim of which is to identify the most distinct subset of features in the whole feature set and thus provide a suitable recognition rate for a particular classifier [5].

Traditional feature selection methods can be divided into three main categories: filter, wrapper and embedded algorithms. In filter algorithm, the feature selection phase is carried out independently of the training learner phase. The method uses traditional information theory, chi-square tests, mutual information and correlation coefficients to make a rough selection of features. For example, the Mutual Information based Feature

Selection method [6,7] and the Conditional Mutual Information Maximization [8] etc. These methods propose to pre-process the feature set and filter out the least relevant features to reduce the feature set's dimensionality, but due to the complexity of the formulae and the high time, complexity can only be used for smaller data sets and pre-classification. Wrapper algorithm differ from filter algorithm in that the performance results of the learner determine the selection of a subset of features. Embedded algorithm complete feature selection and learner training in the same optimization process. Wrapper algorithms select features mostly in conjunction with machine learning classifiers and intelligent algorithms. In this paper, the meta-heuristic algorithm decides whether a feature in the full set of features is added to the feature subset, and the performance of the algorithm determines the number of features selected, so the use of the algorithm to determine feature selection belongs to the wrapper algorithm.

Assuming that there are n features in the feature set, the search space for feature selection is 2^n . Feature selection being an NP-hard problem, traversing it to get all possible solutions is impossible in some cases [9]. Meta-heuristics have the advantages of high efficiency, superiority, and robustness compared to the traditional greedy algorithm hill-climbing algorithm and the ability to obtain a solution or several near-optimal solutions within a sufficient space and time scale, and are therefore increasingly used by researchers to solve complex optimization problems. As part of the meta-heuristic algorithm, the evolutionary algorithm is inspired by the phenomenon of biological evolution in nature. Dr. Holland first proposed the genetic algorithm in 1975 [10], which follows Darwin's 'survival of the fittest, natural selection' law of evolution, whereby populations are renewed by three leading evolutionary operators: selection, crossover, and mutation. However, in nature, species reproduce not only sexually in pairs but also in a few species that have only one parent and do not require gametes. The brood itself does not combine with sex cells to produce offspring such as lower multicellular animals, unicellular plants, algae, ferns, fungi, and bacteria. These species are characterized by a small number of species and a single community that can reproduce in a short period. However, this is both an advantage and a disadvantage; when there is a sudden change in the environment, the community's organisms die off in large numbers, indicating that the populations produced by asexual reproduction are not well adapted. In 2009, J Cantó et al. [11] proposed an asexual genetic algorithm for solving complex mathematical function maximization problems with two variables and optimizing the parameters of the chi-square test in model fitting. The algorithm does not require crossover operators to generate offspring, and offspring are renewed like the way bacterial cells are divided by a single parent randomly selecting different points within a narrow domain, and both parents are always retained if they are more suitable than the offspring. Experimental results showed that this codeless asexual genetic approach is more efficient than traditional genetic algorithms in solving continuous optimization problems, and it is also computationally cheap and requires fewer generations to reach a global solution. In 2010, Alireza Farasat et al. [12] built mathematical models based on the budding mechanism of asexual reproduction to solve optimization problems and decision problems. The experimental results proved the convergence of the algorithm. They verified that the asexual reproduction algorithm has excellent advantages in solving real-time decision problems by exploring the search space without limiting the convergence time and has superior performance compared to these swarm intelligence algorithms of PSO. In 2013, Anabela Simões et al. [13] proposed an asexual permutation genetic algorithm inspired by the DNA sequence structure discovered by Barbara McClintock in the 1950s. Unlike the simple permutation of a sexual mechanism, the asexual permutation genetic algorithm has single parents, while transposons and insertion points are made on a single individual. The genetic algorithm is compared to genetic algorithms with single, multiple, and random crossover operators and finds that it always finds better optima than other sexual reproduction algorithms for both large and small populations. In 2015, Mehrdad Amirghasemi et al. [14] proposed an effective asexual genetic algorithm to solve JSP problems, which combined asexual genetics, an elite pool, and tabu search, with the biased

mutation to increase the diversity of the search space and update to the elite pool used to balance exploration and exploitation. The results also demonstrated the effectiveness and efficiency of using this asexual genetic algorithm in solving JSP problems.

The advantages of asexual genetic algorithms in solving optimization problems in different domains have been reviewed above. The genetic algorithm itself, a discrete coding approach, is naturally well equipped to solve feature selection problems, and there is a dearth of research on single asexual genetic algorithms for FS problems, so the following analysis is given in this paper to demonstrate that the algorithm performs equally well in solving FS problems. The main contributions of this paper are concluded as follows:

(1) The effect of population size on the genetic algorithm was verified.
(2) There is no crossover operator in the asexual genetic algorithm, so this paper verifies the effect of different mutation probabilities on the algorithm foe feature selection.
(3) The performance of the improved genetic algorithm is demonstrated in Android malicious application detection.
(4) The improved genetic algorithm is implemented for feature selection.

The rest of this paper is organized as follows. Section 2 introduces background and method. Section 3 proposes the algorithm of this paper. Section 4 applies the algorithm to the Android malware detection problem. Section 5 demonstrates the effectiveness of the algorithm proposed in this paper through experiments. Section 6 summarizes the article.

2. Background and Method

2.1. Feature Selection

The feature selection problem differs from traditional optimization problems. It is identified as a discrete binary problem where the search space is an n-dimensional lattice space of Boolean type, and the solution to feature selection is to display and update at each corner of the hypercube [15].

$$X_i = (x_{i1}, x_{i2}, ...x_{iD}),\ x_{ij} \in \{0, 1\} \tag{1}$$

where $x_{ij} = 1$ represents the j-th feature is selected into the i-th feature subset x_i, whereas $x_{ij} = 0$ means this feature is not selected.

Thus, the feature selection problem can be formulated as the following optimization problem:

$$\begin{cases} max\ f(X) \\ s.t. X = (x_1, x_2, ...x_D), x_i \in \{0, 1\} \\ 1 \leqslant \{X_i\} \leqslant D \end{cases} \tag{2}$$

where $\{X\}$ represents the set of selected feature subsets, i.e., a subset of features. $f(X)$ denotes the fitness of the selected feature subset X, which is the accuracy of the classification.

2.2. Genetic Algorithm

The purpose of feature selection is to select some of the most compelling features from the original features in order to reduce the dimensionality of the dataset. A subset of the selected features will result in higher classification accuracy [9]. As a class of optimization problems, many researchers have applied evolutionary algorithms and swarm intelligence optimization algorithms to this. The particle swarm optimization algorithm is a collaborative group-based search algorithm that simulates the foraging behavior of a flock of birds, using the individual extremum *pbest* and the group extremum *gbest* to find the algorithmic optimum (solution) [16,17]. The artificial bee colony algorithm is an optimization algorithm based on the honey bee colony's honey harvesting behavior [18]. The grey wolf optimization algorithm, inspired by the predatory behavior of grey wolf packs, finds optimal solutions through collaboration between wolf packs [19,20]. The pigeon flocking algorithm simulates pigeon homing behavior with minimal adjustment parameters and is easy to implement [21]. Genetic algorithm, one of the most classical

evolutionary algorithms, is a learning method inspired by biology. It is a random search and optimization algorithm. Genetic algorithms balance the exploration and exploitation of algorithms through three major evolutionary operators: selection, crossover and mutation, preventing premature maturation of the algorithm and thus finding the optimal solution. It has now been widely used in various fields, such as circuit wiring problems, task scheduling, and machine learning classification tasks. For example, in 1998 Jing-Wein Wang et al. [22] proposed the use of a genetic algorithm as an evaluation function for feature selection, which largely improved the performance of the selected subset of features. The algorithm searches a huge candidate object space and finds the best performing objects according to the fitness function. Individuals with good fitness will be retained during the iteration. Flawed individuals are eliminated or selectively mutated to enter the next iteration. As the iterative process increases, the initial population is updated until the termination conditions are satisfied, or a certain threshold is reached to obtain the final individuals. The flow of the above algorithm to solve the FS problem is shown in Figure 1.

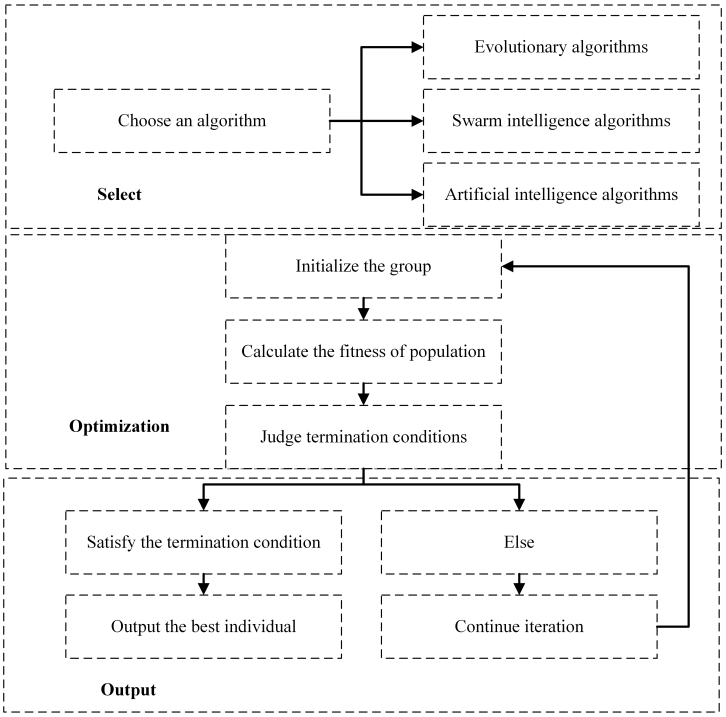


Figure 1. Intelligence algorithm optimization model.

The standard genetic algorithm simulates the evolution process of natural organisms, and the selection operator embodies the environmental selection process of "natural selection, the survival of the fittest" in the evolution of organisms. The crossover and mutation operators play a key role in population renewal during the iterative process. The crossover operator simulates the mating process of individuals in nature, thus increasing the population diversity and theoretically improving the global search capability of the algorithm; unlike the crossover operator, the mutation operator simulates the genetic mutation of individuals in the population by mutating a gene position of an individual (chromosome) from the individual itself, thus improving the local search capability of the algorithm.

2.3. Random Forest Algorithm

The random forest algorithm is part of a large branch of machine learning currently known as ensemble learning. As the name suggests, the algorithm is derived from the decision tree algorithm, where a number of weak classifiers make predictions and then the final strong classifier gives the result. The more correlated any two trees in the forest are, the greater the error rate, while the stronger the classification ability of each tree, the

lower the error rate of the whole forest, i.e., the random forest integrates the advantages of each tree's classification result. The classifiers before the fourth section all use the random forest algorithm.

3. The Proposed Self-Variant Genetic Algorithm (SV-GA)

3.1. Theoretical Basis

In general, the feature selection problem is encoded in binary, and even if the feature values correspond to real values, they are mapped to binary space when solving the problem.To theoretically demonstrate the redundancy of the crossover operator, we first simulated the process of chromosome change in the algorithm.

A population $M(t)$ with n chromosomes is randomly generated, and each chromosome is composed of a string p of size N. Each position i in the string represents the locus of each chromosome, $p\_i = 1$ means that the feature is selected, $p\_i = 0$ means that the feature is not selected, that is, the current gene position is assigned a value of 1, and the number of 1 in the chromosome gene position represents the number of features carried by the current chromosome. The chromosome initialization code is shown in Figure 2.

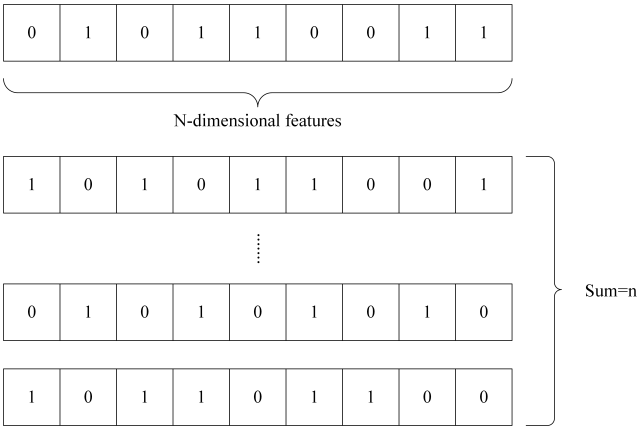


Figure 2. Feature encoding method.

To better understand the advantages of asexual genetic algorithms in feature selection, this paper proposes to analyze its principles by fictionalizing different individuals.

Individual 1 with random 0–1 sequence, as shown in Figure 3:

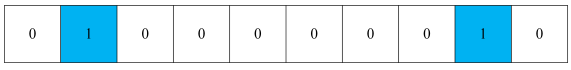


Figure 3. Individual 1.

Individual 2 with random 0–1 sequence, as shown in Figure 4:

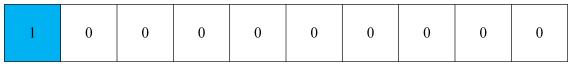


Figure 4. Individual 2.

Individual 3 with random 0–1 sequence, as shown in Figure 5:

Figure 5. Individual 3.

In a standard genetic algorithm, two different or identical chromosomes can be crossed by a single point to obtain a new and different chromosome from the parent, that is, randomly find a cut, exchange its head or tail to obtain a new individual, and reproduce

the process biological reproduction. The single-point crossing process of the above two benign individuals is as follows:

The single point of intersection of individual 1 and individual 2 is shown in Figure 6:

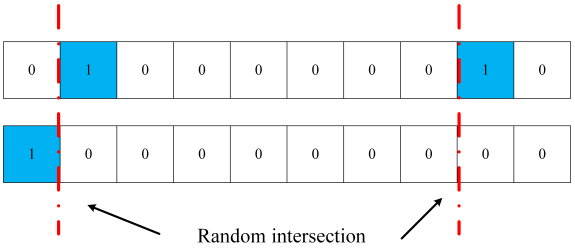


Figure 6. Schematic diagram of single point crossing.

Get new offspring individuals 3, 4, as shown in Figure 7:

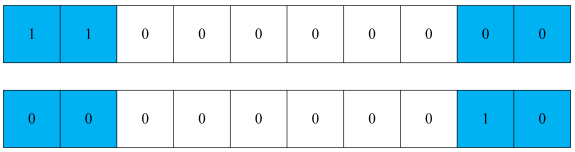


Figure 7. Crossingto produce new individuals.

The Individual 2 mutates to obtain the Individual 3 situation, as shown in Figure 8:

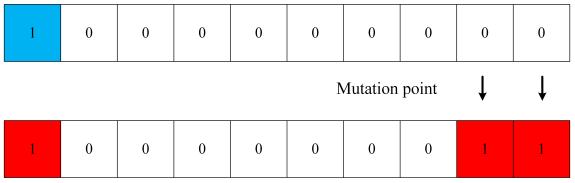


Figure 8. Schematic representation of the changes in individuals after mutation.

From the above crossover variation process, it is clear that the paternal chromosome crossover may only change the value of one gene locus, with no change in the rest of the gene locus other than the crossover point, and the same is true for two-point and multi-point crossovers, where only the value of the gene near the crossover point changes.

3.2. Fitness Function

In machine learning classification tasks, the accuracy of the classification is usually used as a fitness function. Still, for feature selection problems, the fitness function is determined not only by considering the classification accuracy but also by taking into account the number of feature subsets. When different algorithms have the same accuracy, it is better to choose the algorithm with fewer features. The fitness function is defined as follows:

$$Fitness(X_i) = f(L_i), f(L_i) = \frac{correctly\ predicted\ samples}{total\ number\ of\ samples} \tag{3}$$

$$Fitness = \alpha f(L_i) + \beta \frac{n}{N} \tag{4}$$

where X_i and L_i are the i-th individual and corresponding feature subset, $f(L_i)$ is the accuracy of the random forest classifier, n is the number of features selected, and N is the number of features in the dataset, where α usually takes 0.99 and β takes 0.01 as know in [23].

3.3. The Algorithm Flow of the SV-GA

According to the above description, we can find that in solving the feature selection problem, what affects the accuracy of the classifier is the number of selected feature bits

in the individual, i.e., the number of sign bits of 1. Then the corresponding ones in the genetic algorithm are the gene bits of each chromosome. When the chromosome undergoes mutation, each gene bit can be operated by changing 0 to 1 and 1 to 0. At this point we can roughly assume that a single mutation operation can satisfy the needs of the features when performing classification. The steps of its application in feature selection are as follows:

Step1: Initialize the population N, t is the number of current iterations, and the total number of features is P.

Step2: Calculate the fitness value of individuals in the current population, and obtain the fitness value $F(n)$ of each chromosome after n calculations, that is, obtain the fitness value of each individual with different characteristics.

Step3: Selection. In the SV-GA algorithm, the selection method is tournament selection, and the number of individuals selected each time is 3, and the tournament selection method is replacement sampling. Three individuals are randomly selected from the population to calculate their fitness, and the better individual directly enters the next generation.

Step4: Mutation. According to a certain mutation probability, a mutation operation is performed on each chromosome in the population, and the mutated individuals differ from those after the selection operation and carry different characteristics, and these individuals with different characteristics constitute a new population.

Step5: Algorithm termination conditions. If it satisfies the artificially set number of iterations, the algorithm terminates, set the number of iterations then output the feature subset selected by the algorithm and skip to *Step6* at the end of the iteration. If not, then execute Step2.

Step6: Output results. The individual with the highest fitness value is output and the feature with a gene position of 1.

3.4. Computational Complexity Analysis

In SV-GA, the factors affecting the time complexity of the algorithm are not only the population size N, but also the number of iterations t. The feature selection problem, as a class of optimization problems, aims to improve the classification accuracy by reducing the number of features for training, and the dimensionality D of the features in the sample set also affects the efficiency of the algorithm in the process of algorithm optimization. In summary, the time complexity of the algorithm can be summarized as $O(t * N * D)$, and the number of individuals in the population and the feature dimension are the main factors affecting the computational complexity.

3.5. Numerical Analysis

The SV-GA changes the iterative process of the traditional genetic algorithm, and increases the diversity of individuals in the population by changing the population size. The impact of mutation operations on the performance of the algorithm should not be underestimated. In this section we first verified the effect of different mutation probabilities on the algorithm, comparing the performance of GA and SV-GA, and then we verified the effect of different population sizes on the algorithm. All results are mean results from 10 independent runs with 100 iterations.

3.5.1. Datasets

To verify the effectiveness and applicability of the algorithms, this paper uses SV-GA to test on different UCI standard binary classification test datasets [24]. Table 1 provides the name of the dataset, the number of samples contained in the dataset, the total number of features, and the number of features selected by different algorithms.

Table 1. Dataset Properties.

No.	Dataset	Number of Samples	Number of Features
D1	SPECT Heart	267	22
D2	CMC	962	9
D3	Sonar	207	60
D4	Credit6000	6000	65
D5	Heart-statlog	270	13
D6	Spambase	4600	57

3.5.2. Results with Different Mutation Rate

In the meta-heuristic algorithm, the search operator will deal with the overall selection pressure, convergence problem, randomization, and diversity, which are all dedicated to exploration and exploitation [25]. Based on the characteristics of the feature selection problem, we retain the selection and mutation operators in the SV-GA algorithm. On the one hand, we change the number of initial population sizes to increase the diversity of the population, i.e., to improve the global search capability of the algorithm, and on the other hand, we experimentally verify the effect of the mutation rate on the accuracy while keeping the population size constant. This subsection is to verify the effect of mutation probabilities on the performance of SV-GA.

To ensure that the variables are unique, the population size for the results of the iterative curve shown in Figure 9 is set to 30 (pop = 30). The results on the six datasets show that the algorithm performs poorly when the mutation rate is set to 0.01, 0.02 and 0.6. The results on all six datasets are the worst when the rate is 0.01. It is not difficult to analyze that the performance of the genetic algorithm with the crossover operator omitted depends on the magnitude of the variance, and when the variance is set to 0.01, the algorithm is close to 0 variance, i.e., the features involved in the classification are initialized features, the structure of the population does not change significantly and the algorithm falls into a local optimum solution. Overall, on both the SPECT heart and the Spambase, the algorithm performed optimally when the probability of variation of the algorithm was set to 0.2, when the population was best adapted.

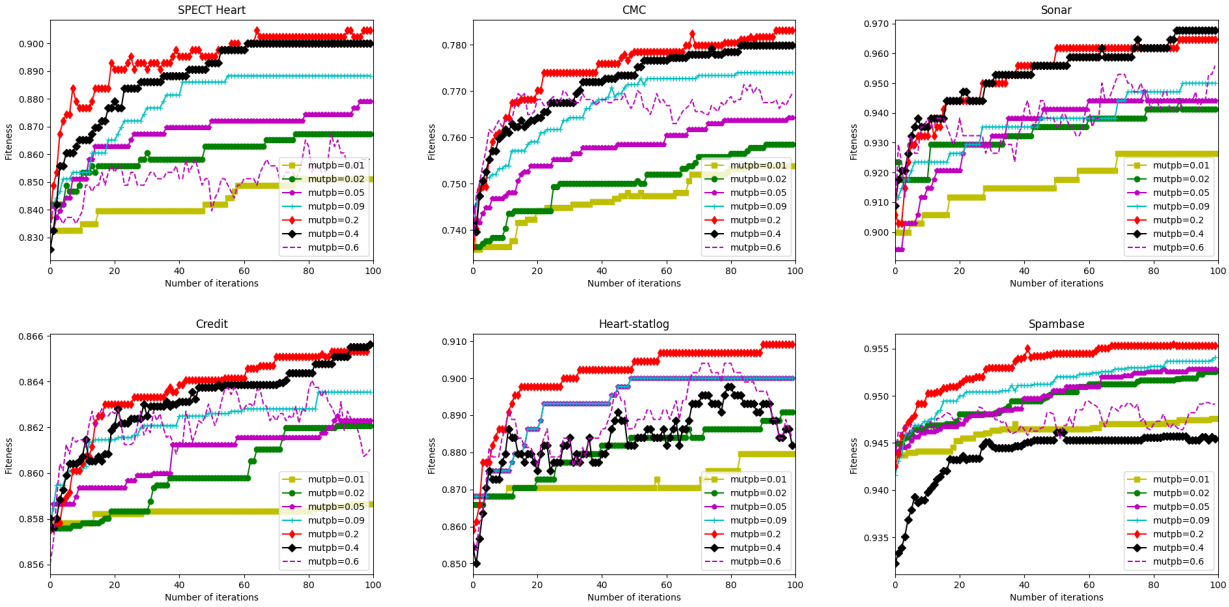


Figure 9. The evolutionary curves of different mutation rate for datasets.

3.5.3. Comparison between GA and SV-GA

Table 2 shows the parameter settings for GA and SV-GA when comparing experiments on different UCI datasets.

Table 2. Parameter settings.

Algorithms	Parameter	Value
GA	population	30
	Selection	Tournament Tournize = 3
	Crossover	One-point Cxpb = 0.5
	Simple Mutate	Mupb = 0.2
SV-GA	population	30, 50, 100
	Selection	Tournament Tournize = 3
	Simple Mutate	Mupb = 0.2

The iteration curves shown in Figure 10 show a comparison of the algorithms for GA and SV-GA. The algorithms were both iterated 100 times, with a population size of 30 for both GA and SV-GA and a mutation probability of 0.2, with a crossover probability of 0.5 for the former. From the experimental results, it is easy to see that the fitness value of SV-GA is increasing with the number of iterations. The effect is more evident on the SPECT Heart, CMC, Sonar, Credit6000, and Spambase datasets, while in Heart-statlog, the fitness value of the algorithm is unstable if not as good as GA. Still, the algorithm later convergence is faster than the former. It is easy to see from the graphs of the iterations of the two algorithms on different datasets that the asexual genetic algorithm has an advantage in solving the feature selection problem.

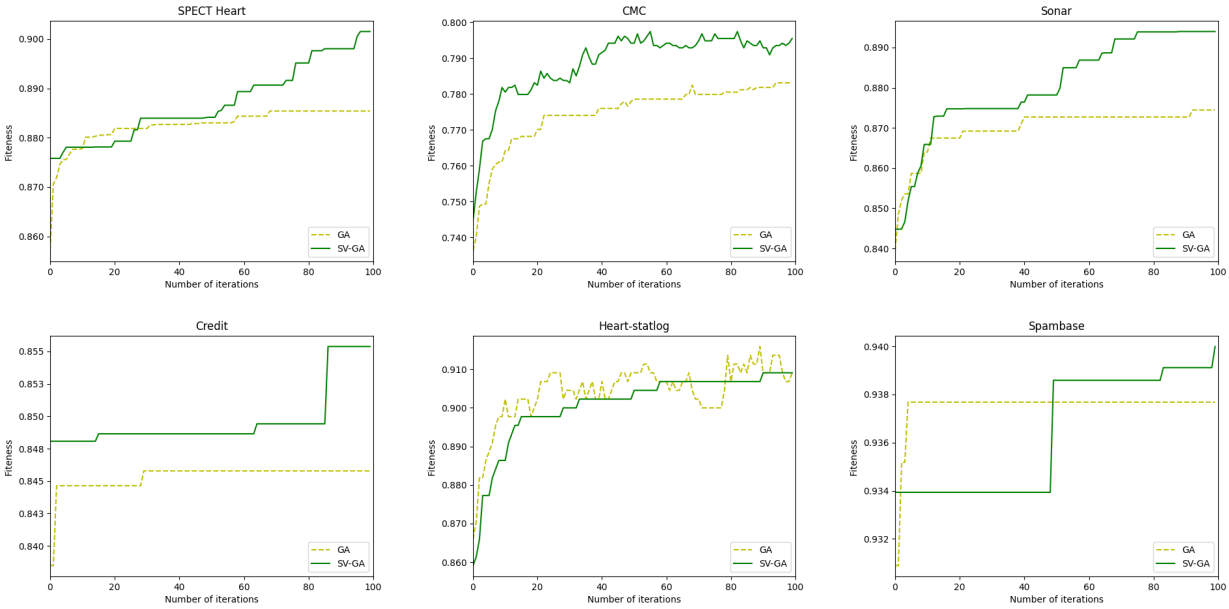


Figure 10. The evolutionary curves of GA and SV-GA.

3.5.4. Symmetry Theory and Algorithms

Symmetry is defined in both mathematics and physics. Mathematically, symmetry is defined as graphical symmetry and numerical palindromes; physically, symmetry is defined as invariance after some operation, both in time and space. In the asexual genetic algorithm proposed in this paper for solving the feature selection problem, the population size does not change due to a change in strategy. For example, if the number of individuals in the initialized population is 30, then as the number of iterations increases, the number of individuals remains at 30. The optimal solution is the best value (maximum or minimum)

among the 30 individuals when solving the optimization problem. Therefore, in the following subsection, we verify the algorithm results for population sizes of 30, 50, and 100, respectively.

3.5.5. Results of the Population Size

A comparison of the fitness values of the standard GA with those of the SV-GA algorithm for different population sizes is shown in Table 3, where the SV-GA algorithm selects the operator for tournament selection and the mutation probability is set to the best mutation probability of 0.2 as experimentally demonstrated above. Analysis of the values in this table shows that the algorithm with a population size of 100 performs better than the algorithm with population sizes of 30 and 50 while keeping the variation probability constant.

Table 3. Mean and standard deviation results of acc on fitness and its competitors for six Datesets.

	GA		SV-GA(30)		SV-GA(50)		SV-GA(100)	
	Mean	**(Std)**	**Mean**	**(Std)**	**Mean**	**(Std)**	**Mean**	**(Std)**
D1	0.865	(0.006)	0.882	(0.003)	0.887	(0.008)	**0.897**	(0.006)
D2	0.752	(0.008)	0.747	(0.004)	0.748	(0.006)	**0.755**	(0.007)
D3	0.868	(0.015)	0.871	(0.012)	0.882	(0.013)	**0.887**	(0.013)
D4	0.846	(0.002)	0.849	(0.001)	0.849	(0.002)	**0.850**	(0.002)
D5	0.891	(0.018)	0.894	(0.020)	0.898	(0.016)	**0.904**	(0.012)
D6	0.938	(0.002)	0.939	(0.001)	0.940	(0.002)	**0.943**	(0.002)

The reason for this is that the local search capability of the algorithm is stable when the variation probability is constant, while the more individuals within the population, the greater the diversity of the population, i.e., the greater the global search capability of the algorithm. By varying the population size and variation probability, the SV-GA balances the exploitation and exploration capability of the algorithm, thus facilitating the algorithm to find the global optimal solution.

4. Android Malicious Application Detection Based on SV-GA Algorithm

Feature selection is an important step in processing classification tasks and in the preprocessing phase of data mining, with the aim of improving the accuracy of the classifier. In order to verify the performance benefits of the proposed algorithm, we apply it to the Android malicious application detection problem and propose a framework process to solve the problem.

4.1. Android Malicious Application Detection

With the continuous advancement of time and the increase in the number of third-party application markets, many researchers have explored the detection methods of Android malicious applications from a software perspective to achieve the purpose of protecting system stability. In this paper, we focus on solving the security problem at the application level of Android by compiling the corresponding feature sets based on the source code information obtained by the decompiler tool and feeding them into the classifier model for training to classify benign and malicious applications. Due to the high dimensionality of the acquired feature attributes, it inevitably causes a dimensional disaster or increases the time and space complexity of model training, which affects the efficiency of Android malicious application detection. It is vital to choose a suitable feature selection method to reduce the number of features while improving the accuracy of the classification model.

Malicious application detection methods are frequently updated and changed with the deepening of research. The main research trends are in the following two aspects: one is based on the improvement of a single machine learning classification algorithm; the other is based on the study of feature selection methods. Machine learning methods are currently

the most widely used technical means in the artificial field. The classification methods in supervised learning play a driving role in the detection of Android malicious applications (such as the KNN algorithm, Naive Bayes algorithm, Logistic Regression, and Decision Tree algorithm). The disadvantage is that the training model speed needs to be improved, and it can only make a simple judgment of the malicious software that has already appeared on the market and cannot realize the detection of unknown types of applications. Feizollah A, Nor B, Salleh R et al. [26] evaluated the performance of K-means and Mini batch K-means clustering algorithms in Android malware detection and analyzed the network traffic of benign and malware on two algorithms. The result showed that the overall performance of the Mini batch K-means clustering algorithm was better than the K-means algorithm. Nath, Hiran V et al. [27] applied the classification algorithm in machine learning to features such as n-gram model and byte sequence extracted from malware. The algorithm included classification methods such as decision trees and boosted decision trees. The results proved the machine learning classification algorithm could realize the simple classification of malware. However, only using machine learning algorithms to judge the quality of the application is slightly thin. Rajesh Kumar et al. [28] proposed a method based on the combination of probabilistic statistical analysis and machine learning algorithms to reduce the dimensionality of features and achieved classification between known and unknown benign and malicious software.

Android malicious application detection based on feature selection method is divided into static analysis and dynamic analysis methods [29]. Static analysis involves obtaining the source code of an Android application by decompiling software without running the application, analyzing it to extract relevant syntactic and semantic information, permission information in configuration files, intent, the corresponding API calls, etc., coding and mapping its code integration into vector space, and combining it with machine learning classification in order to achieve malicious application classification. In contrast, dynamic analysis is similar to the black-box testing of software. The source code structure is not taken into account, and only relevant features are obtained during the installation or use of the application, such as network traffic analysis, application power consumption, user behavioral features. Dynamic analysis has the advantage of a large feature selection space and a wide range of input classifiers. For example, Zarni Aung et al. [30] extracted single permission as a feature for training, designed and implemented a framework based on machine learning technology classify malware and benign software. Shanshan Wang et al. [31] conducted an in-depth study on the behavior of network traffic generated by the application during use, mapped the mobile terminal traffic information flow to the server-side, analyzed the network traffic characteristics, and combined the C4.5 algorithm to complete the detection of malicious applications. Du W, Yin H et al. [32] proposed a method to describe Android malware that relied on API calls and package-level information in bytecode and determined the category of unknown application software based on known Android malicious applications. Compared with the classifier based on permission features, the KNN classifier's accuracy was as high as 99%. Daniel Arp et al. proposed a lightweight detection framework: Drebin [33]. This method extensively collected application characteristics (permissions, hardware combinations, etc.) obtained from static analysis and mapped them to the joint vector space, using traditional PCA to reduce dimensional method selection features. The biggest advantage of this framework is the ability to identify malicious applications on smartphones directly. Wang W, Gao Z, Zhao M et al. [34] proposed an Android malicious application detection model: DroidEnsemble. The classification features in the model analyze static string features such as permissions in each application code pattern and include structural feature s such as control flow graphs and data flow graphs, such as function call graphs. The results of classifying these two types of features show that the model's detection accuracy is greatly improved, while the false alarm rate is also reduced. The approach based on feature selection has certain advantages, but it can lead to the high dimensionality of the feature combinations, leading to the high complexity of the algorithm's training process in space and time and affects the accuracy of the machine

learning and data mining methods. Therefore, the selection of features with good detection performance is key to the method.

4.2. Construction of Feature Sets

Feature selection aims to reduce the number of features used for classification while maintaining classification accuracy [35]. Based on the dynamic and static analysis mentioned above, we have chosen the static analysis method to classify the software. One is that the static analysis method is not only simpler but also less harmful to mobile devices. When using a mobile device, the application will request a permission to respond during installation and the system will simultaneously check whether the permission is invoked. Permissions become one of the indispensable static features for detecting malicious Android applications [36].

4.3. APK Pre-Processing

The Android application package (APK) of the third-party application market is not presented in source code but is similar to the packaged file format (zip). In order to obtain the information in the package, it is necessary to use a decompiler tool to realize the pre-processing of APK decompression. The tool used in this article, Apktool [37], is a lightweight decompilation tool, a closed binary Android application tool, which can decode resources and applications into the most primitive state of java source code, and automatically realize file structure processing. It can be used locally and supports multiple platform analysis applications. As shown in Figure 11 is the file resource list obtained by decompiling a real Android application using this apktool tool.

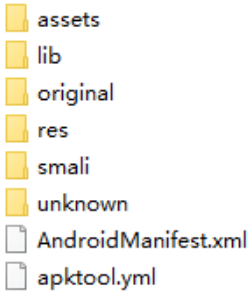


Figure 11. Example of file resource list.

After decompiling the APK, it can see the permission information in the total configuration file Androidmanifest.xml. Figure 12 shows a partial list of permissions for a test APK. P is the total set of possible permissions that some applications in the android platform can request. Moreover, each android application is represented in the framework of the required permission set. Therefore, assuming that the size of P is N, each application is represented by a binary string p of size N, where each position i of the string represents the i-th permission in a set of possible permissions, so that $p\_iin0, 1$. If the application does not need permission i, then $p\_i = 0$; if the application requires permission i, then $p\_i = 1$.

```
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.GET_ACCOUNTS"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.RECORD_AUDIO"/>
<uses-permission android:name="android.permission.VIBRATE"/>
```

Figure 12. Example of file resource list.

4.4. Coding

When classifying Android malware, permission features are binary coded in such a way that if total permission sets of Android applications are defined as P, then the

permissions applied by each software are $N(N <= P)$, coded one by one, and if a feature in permission set P appears, then a permission bit in N is flagged as 1, otherwise it is 0. An example of matching permissions for a single application is shown below :

0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,1,0,0,1,0,0,0,0,0,0,1,0,0,1, 0,0,0,0,1,0,1,1,0,0,0,1,1,0,0,0,0,0,1,0,1,0,1,0,0,0,0,0,1,0,1,1,0,0,1,0,1

There are 88 permission bits in this example. The exact number of settings is described in the numerical experiments section. When each benign application is developed, the permissions applied are very few (compared to the official permission set). The number of "1" s in the gene position of each chromosome is relatively small. The number of permissions requested by malicious applications is greater than or equal to that of benign applications.

4.5. System Framework

This paper proposes a system model based on the permissions requested by Android applications, including three modules: decompilation, feature selection, and application classification. The block diagram of Android malicious application detection is shown in Figure 13. The proposed model consists of the following modules:

1. Decompilation module. Use the decompilation tool "Apktool" [37] to decompress each application package file to obtain the total configuration file Androidmanifest.xml of the application and obtain the permission list for each APK. All the extracted permissions are used as the original feature set.
2. Feature selection module. This module optimizes the original data set and selects the feature subset that dramatically impacts the classification effect. The original subset is randomly selected as the initial iterative group, then iterated through the improved genetic algorithm to screen the group, and finally, get the set of permission features that optimize the classification effect.
3. Classification module. The feature data set extracted by the decompiler module is selected by an asexual genetic algorithm to obtain a feature subset, which is fed into a machine learning classifier for training, and its classification accuracy judges the feature selection method to achieve the classification of Android software.

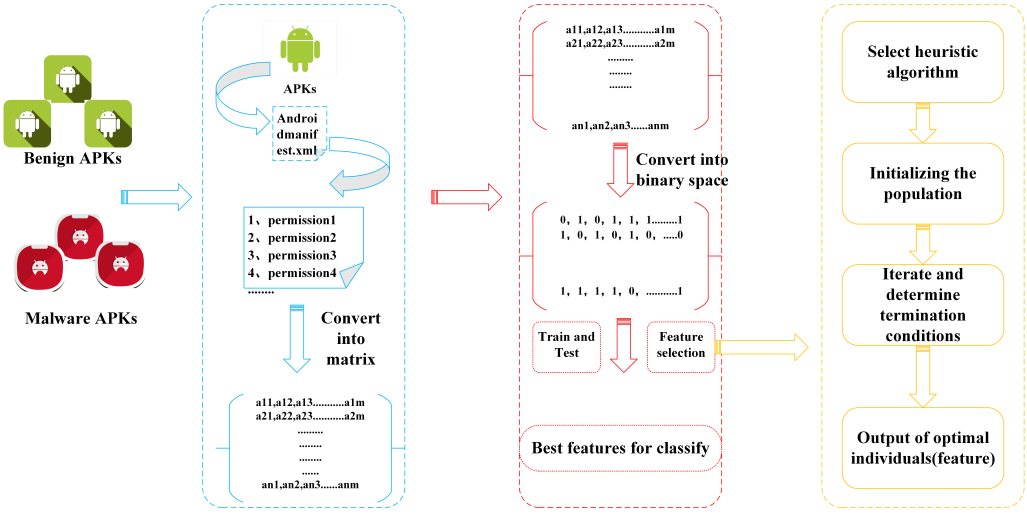


Figure 13. System model diagram.

5. Experimental Research

In this subsection, the feasibility of the asexual genetic algorithm for binary feature selection is further demonstrated with experimental data using different data sets and comparisons between different feature selection methods.

5.1. Numerical Verification

There are 1788 benign samples and 932 malicious samples in the experimental data of this paper. Among them, 532 malicious samples are from the Canadian Institute of Cyber Security [38], the samples contain typical Android platform malware, such as malicious ransomware applications, threatening SMS applications, and advertising applications; there are also 400 malicious samples from the data set of Dr. Wang's repository (http://infosec.bjtu.edu.cn/wangwei/?page_id=85 (accessed on 17 July 2020)). Total 188 benign Android applications, mainly from Google play and Xiaomi App Market. After security testing, these apps are released to the app market and are therefore deemed benign and safe. Besides, there are 1600 benign samples from Dr. Wang's database.

5.1.1. Permission Description

The number of permissions matched by the samples from Canadian Institute, Google play market, and Xiaomi app market is the official 144 permissions, while the number of permissions given by Dr. Wang's dataset is the most frequently selected 88 permissions after filtering, so the number of permissions for unified features in this paper is set to 88. The 88-feature datasets are input to logistic regression (LR), random forest (RF), Gaussian Naive Bayes (GNB) and K-nearest neighbor (KNN) classifiers for training and testing, respectively.

5.1.2. Classifier Model Evaluation

The primary objective when classifying Android applications is to flag malicious applications from the list, as only malicious software poses a risk to the system and user security, and not all software can be correctly classified when the classifier is trained, so we need an confusion matrix to describe the different types of errors and measure the severity of the errors separately. In the confusion matrix, as shown in Table 4, each column represents the instances in a predicted class, and each row represents the instances in an actual class. A specific table layout allows visualization of the performance of the classifier [17]. In this article, the positive class represented malicious applications, and the negative class represented benign applications.

Table 4. Classification results of the four classifiers under 88 permission features.

	Malware Class	**Benign Class**
Malware prediction	TP	FP
Benign prediction	FN	TN

Let *TP* (True Positive) be the number of malicious applications correctly predicted as Android malware. *FP* (False Positive) be the number of benign applications incorrectly predicted as Android malware. *FN* (False Negative) is the number of malicious applications that are incorrectly detected as benign applications, and *TN* (True Negative) be the number of benign applications correctly detected as benign applications. The following is the evaluation standard formula of the classifier:

Sample Accuracy (*ACC*) represents the percentage of the overall data set that is correctly classified. The higher the *ACC* value, the better the classification effect. It is defined as follows:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

True positive rate (*TPR*) represents the probability that a positive sample will be correctly predicted as positive. The higher the value, the more effective the classifier is. It is defined as follows:

$$TPR = \frac{TP}{TP + FN} \tag{6}$$

False Positive Rate (*FPR*) represents the probability that a negative sample is falsely predicted to be positive. The higher the value, the worse the effect of the classifier. It is defined as follows:

$$FPR = \frac{FP}{FP + TN} \tag{7}$$

Precision (*P*) represents the probability that a positive sample is predicted to be positive, and is defined as follows:

$$P = \frac{TP}{TP + FP} \tag{8}$$

Recall is the probability that a positive sample is predicted to be positive, and the equation is the same as the true rates. It is defined as follows:

$$Recall = \frac{TP}{TP + FN} \tag{9}$$

F1-score ($F - S$) This indicator considers both precision rate and recall rate, so that both reach the highest at the same time, defined as follows:

$$F - S = \frac{2 \cdot P \cdot Recall}{P + Recall} \tag{10}$$

In order to find a classifier that makes the best classification performance among many machine learning classification models, this paper uses unselected (unpreprocessed) datasets of features input to different classifiers, relying on the goodness of the above classification metrics to determine the classification model applied in the following comparison experiments. The experimental results are shown in Table 5.

Table 5. Classification results of the four classifiers under 88 permission features.

No.	Classifier	*ACC*	*TPR*	*FPR*	$F - S$
1	LR	0.901	0.833	0.097	0.372
2	RF	0.867	0.958	0.136	0.337
3	GNB	0.955	0.958	0.136	0.337
4	KNN	0.933	0.208	0.018	0.247

The purpose of feature selection is to improve the classification accuracy while reducing the number of features for training the classification model; therefore, high precision and high true rate become the indicators for judging the good and bad pairs of classification models. The experimental results show that the GNB has the highest ACC and TPR and the model's best overall performance when classifying the dataset without feature selection.

5.2. Discrete and Continuous Algorithms

In a feature selection optimization problem, the search space is a hypercube, and all solutions lie only at values of 0 or 1 [20]. Genetic algorithms do not require changes to them in solving FS problems due to their natural discrete encoding. In contrast, in swarm intelligence algorithms such as GWO, WOA, and ACO, where the search range of the algorithm is a continuous space, many researchers have proposed a binary form of the algorithm in order to enable optimization of discrete problems. Kennedy proposed a binary version of PSO in 1997 [39], and much work has been done around this version. The algorithm uses a sigmoid function to map a vector in continuous space to a two-dimensional space, and the mapping equation is shown below.

$$S\left(X_{ij}^{t}\right) = \frac{1}{1 + e^{-X_{ij}^{t}}} \tag{11}$$

$$X_{ij}^{t+1} = \begin{cases} 1 & if\ S\left(X_{ij}^{t}\right) > \sigma \\ 0 & otherwise \end{cases} \tag{12}$$

$$X_{ij}^{t+1} = \begin{cases} \left(X_{ij}^{t}\right)^{-1} & if\ T\left(X_{ij}^{t}\right) > \sigma \\ X_{ij}^{t}\ otherwise \end{cases} \tag{13}$$

In addition to this, four S-shaped and V-shaped mapping functions were introduced by Shahrzad Saremi et al. [40] to transform the continuous space. The equation for the change in X_{ij}^{t+1} when using the S-shaped mapping function is Equation (12) and for the V-shaped mapping function is Equation (13). The mapping functions are shown in the Table 6. Any of the continuous optimization algorithms can achieve binary conversion by means of S-shaped and V-shaped map functions.

Table 6. S-shaped and V-shaped mapping functions.

S-Shaped		**V-Shaped**	
S1	$T(x) = \frac{1}{1+e^{-2x}}$	V1	$T(x) = \left\lvert \frac{\sqrt{2}}{\pi} \int_{-0}^{(\sqrt{\pi}/2)x} e^{t^2} dt \right\rvert$
S2	$T(x) = \frac{1}{1+e^{-x}}$	V2	$T(x) = \lvert tanh(x) \rvert$
S3	$T(x) = \frac{1}{1+e^{-(x/2)}}$	V3	$T(x) = \left\lvert (x)/\sqrt{1+x^2} \right\rvert$
S4	$T(x) = \frac{1}{1+e^{-(x/3)}}$	V4	$T(x) = \left\lvert \frac{2}{\pi} arctan\left(\frac{\pi}{2}x\right) \right\rvert$

5.3. Comparison Algorithm and Their Parameter Setting

The purpose of feature selection is to select a subset of feasible features by eliminating irrelevant, redundant, or noisy features [41]. In order to evaluate the proposed algorithm, a number of existing feature selection algorithms were selected, such as the grey wolf optimization algorithm(GWO) [42], the whale optimization algorithm(WOA) [43] and the ant colony algorithm(ACO) [44]. The methods used in evolutionary algorithms to solve the feature selection problem are as follows:

- Traditional genetic algorithms [10];
- A new binary version of the grey wolf optimization algorithm [23];
- S-type binary whale optimization algorithm [45];
- Ant colony algorithm for binary encoding [46]

In order to be fair, the proposed algorithm is compared with these algorithms under the same parameter settings. The population size for all algorithms is 30, except for the population size for SV-GA, the number of runs is equal to be 10, the iterations is equal to 100. In [46], this paper draws on the parameter settings of α and β in the ant colony algorithm tested in that paper to get the best results, so that $\alpha = 1$ and $\beta = 3$, for comparison with the algorithm in this paper. To verify the performance of the proposed algorithm, we divide each dataset into a training set and a test set, e.g., k-fold cross-validation, dividing the sample set into k subsamples of equal size, selecting k-1 subsamples as the training set and all the remaining subsamples as the test set. Among the many supervised classifiers, we choose the better-performing GNB.

5.4. Experimental Results and Analysis

5.4.1. Fitness Performance

In the simulation experiments section, we compare the SV-GA algorithm with population sizes of 30, 50, and 100 with the GA, GWO, ACO, and WOA algorithms. All experimental data are mean results of independent runs.

Table 7 shows the statistical results using Equation (4) as the fitness function. To show that our proposed algorithm's performance is significantly better than that of the comparison algorithms, we used a non-parametric statistical test: Wilcoxon rank-sum test with a significance level of $\alpha = 0.05$. The null hypothesis is that the proposed algorithm is

not significantly different from the comparison algorithm, and the alternative hypothesis is that the proposed algorithm is significantly different from the comparison algorithm. We use the symbols $+, =, -$ to indicate that the proposed algorithm's performance is significantly better, no significant difference, and significantly inferior to the corresponding comparison algorithm.

Table 7. Comparison of Fitness results on different algorithms for different datasets.

Datasets	GA	SV-GA(30)	SV-GA(50)	SV-GA(100)	GWO	ACO	WOA
	Mean(std)	Mean(std)	Mean(std)	Mean(std)	Mean(std)	Mean(std)	Mean(std)
Android Dataset	0.922(0.004)+	0.927(0.003)+	0.930(0.003)+	0.936(0.002)	**0.947**(0.005)−	0.935(0.004)+	0.925(0.004)+
SPECT Heart	0.865(0.006)+	0.882(0.003)+	0.887(0.008)+	**0.897**(0.006)	0.801(0.033)+	0.830(0.029)+	0.856(0.012)=
CMC	0.752(0.008)=	0.747(0.004)+	0.748(0.006)=	**0.755**(0.007)	0.647(0.028)+	0.644(0.048)+	0.733(0.009)+
Sonar	0.868(0.015)+	0.871(0.012)+	0.882(0.013)=	**0.887**(0.013)	0.755(0.011)+	0.750(0.045)+	0.878(0.020)+
Credit6000	0.846(0.002)+	0.849(0.001)=	0.849(0.002)=	**0.850**(0.002)	0.842(0.004)+	0.774(0.061)+	0.841(0.002)+
Heart-statlog	0.891(0.018)=	0.894(0.020)=	0.898(0.016)=	**0.904**(0.012)	0.709(0.058)+	0.725(0.125)+	0.841(0.029)+
Spambase	0.938(0.002)+	0.939(0.001)+	0.940(0.002)+	**0.943**(0.002)	0.926(0.012)+	0.833(0.047)+	0.921(0.003)+
+/−/=	5/0/2	5/0/2	3/0/4		6/1/0	7/0/0	6/0/1

As can be seen from the table, the fitness value of SV-GA with an initial population size of 100 on the Android, SPECT Heart, CMC, Sonar, Credit6000, Heart-statlog, and Spambase data sets are all higher than those of traditional GA and SV-GA of population size of 30 and 50, the fitness values are 93.6%, 89.7%, 75.5%, 88.7%, 85.0%, 90.4%, and 94.3%. When compared with other algorithms, WOA has the best performance on the Android dataset. In addition, the fitness value of SV-GA with a population size of 100 far exceeds other algorithms. The results of the statistical tests showed that on the Android dataset, SV-GA(100) results were significantly more significant than the other algorithms on the Android dataset, except for the non-significant results compared with the GWO algorithm. On the SPECT Heart dataset, WOA was not significantly different, and the results were significant on the rest of the datasets. On the CMC, Sonar, Credit6000, Heart-statlog, and Spambase datasets, the results were significant compared to the GWO, ACO, and WOA algorithms. Overall, SV-GA(100) ranked first in terms of performance.

5.4.2. Classification and Selected Features

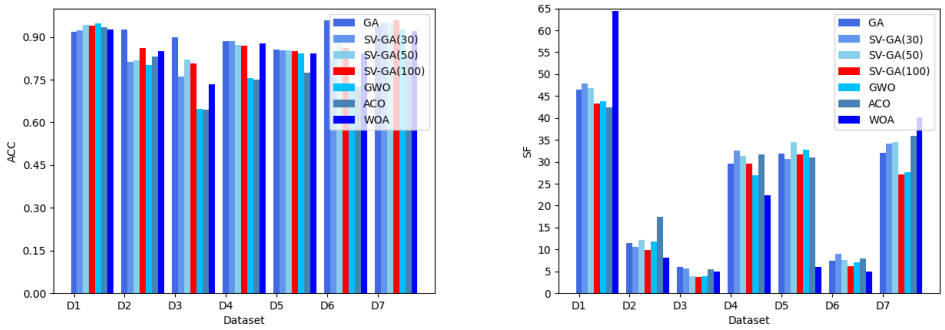
Table 8 shows the results of using Equation (3) as the fitness function. Unlike Equation (4), Equation (3) does not take into account the number of features, and the results of the classifier are used directly as the fitness function. From the experimental results in Table 8, the performance of the genetic algorithm without the crossover operator is worse than that of the traditional genetic algorithm on the SPECT Heart, CMC, Credit6000, and Heart-statlog datasets. The reason for this result is, on the one hand, because of the small number of samples in the dataset and the relatively small number of features in the examples. When no crossover operation is performed, the population's diversity is not guaranteed. The algorithm tends to fall into premature maturity, making it difficult to find the global optimal solution. The results in Table 9 show that on the seven datasets of Android, SPECT Heart, CMC, Sonar, Credit6000, Heart-statlog, and Spambase, the SV-GA with a population size of 100 selected the smallest proportion of features in more than half of the datasets, which is consistent with the goal of optimizing the feature selection problem, i.e., is the accuracy of the classifier is not high, and the number of features selected is low. Still, the final results obtained for both in proportion to specific parameters indicate that the algorithm performance is good. It can be demonstrated that the algorithm has some advantage in solving the FS problem. The graphical results of Tables 8 and 9 are shown in Figure 14a,b. The red bars are the algorithms compared with other algorithms in all experiments.

Table 8. Comparison of ACC results on different algorithms for different datasets.

Datasets	GA	SV-GA(30)	SV-GA(50)	SV-GA(100)	GWO	ACO	WOA
	Mean(std)	Mean(std)	Mean(std)	Mean(std)	Mean(std)	Mean(std)	Mean(std)
Android Dataset	0.917(0.004)	0.923(0.002)	0.941(0.004)	0.940(0.004)	**0.947**(0.005)	0.935(0.004)	0.925(0.004)
SPECT Heart	**0.925**(0.004)	0.811(0.006)	0.817(0.010)	0.860(0.008)	0.801(0.033)	0.830(0.029)	0.856(0.012)
CMC	**0.900**(0.072)	0.762(0.009)	0.821(0.103)	0.806(0.064)	0.647(0.028)	0.644(0.048)	0.733(0.009)
Sonar	0.884(0.014)	**0.886**(0.012)	0.871(0.012)	0.869(0.009)	0.755(0.011)	0.750(0.045)	0.878(0.020)
Credit6000	**0.855**(0.001)	0.853(0.002)	0.854(0.002)	0.849(0.001)	0.842(0.004)	0.774(0.061)	0.841(0.002)
Heart-statlog	**0.958**(0.027)	0.824(0.010)	0.892(0.011)	0.862(0.013)	0.709(0.058)	0.725(0.125)	0.841(0.029)
Spambase	0.936(0.003)	0.950(0.002)	0.947(0.002)	**0.951**(0.002)	0.926(0.012)	0.833(0.047)	0.921(0.003)

Table 9. Comparison of feature number on different algorithms for different datasets.

Datasets	GA	SV-GA(30)	SV-GA(50)	SV-GA(100)	GWO	ACO	WOA
	SF(%)	SF(%)	SF(%)	SF(%)	SF(%)	SF(%)	SF(%)
Android Datase	46.5(52.84)	47.9(54.43)	46.8(53.18)	43.307(49.213)	43.800(49.773)	**42.5**(47.852)	64.400(73.182)
SPECT Heart	11.5(52.27)	10.5(47.73)	12.2(54.45)	**9.9**(45)	11.8(53.636)	17.4(79.090)	8.1(36.818)
CMC	6(66.67)	5.6(62.22)	3.9(43.33)	**3.769**(41.880)	3.9(43.333)	5.5(61.111)	5(55.556)
Sonar	29.6(49.33)	32.6(54.33)	31.3(52.17)	29.6(49.33)	27(45)	31.7(47.692)	**22.4**(37.333)
Credit6000	31.9(49.08)	30.7(47.23)	34.5(53.08)	31.7(48.77)	32.7(50.308)	**31**(47.692)	6.1(9.384)
Heart-statlog	7.368(56.680)	9(69.23)	7.6(58.46)	6.227(47.902)	7(53.846)	8(61.538)	**4.9**(37.692)
Spambase	32.1(56.32)	34.2(60)	32.3(56.67)	**27.176**(47.678)	27.6(48.421)	35.9(62.982)	40.2(70.526)



(**a**) Comparison of accuracy of different datasets (**b**) Comparison of the number of features in different datasets

Figure 14. Comparison between different methods and different datasets on ACC and SF.

5.4.3. Running Time

In this paper, the SV-GA is influenced not only by the initial population size but also by the variation rate. Table 10 shows the mean results for ten independent runs of the different algorithms. It is easy to see from the results that the GA times are much higher than the SV-GA for arbitrary populations. The reason for this result is that the crossover operator increases the algorithm's time complexity. When compared to the other algorithms, GWO takes the least time, which may depend on the co-evolutionary strategy of the algorithm itself to speed up finding the optimal solution. Still, the overall SV-GA time does not pull away from this algorithm by a large margin, and to some extent, there is no significant gap. When solving the feature selection problem, the time complexity depends heavily on the number of samples and the number of features. For example, Credit6000 has the highest number of instances of any of the seven datasets, so no matter how well the algorithm performs, the time on that dataset is bound to exceed that on any of the remaining datasets.

Table 10. Comparison of running-time on different algorithms for different datasets.

Datasets	GA	SV-GA(30)	SV-GA(50)	SV-GA(100)	GWO	ACO	WOA
Android Dataset	102.739	10.195	9.254	9.174	**8.595**	10.265	198.043
SPECT Heart	57.345	4.582	4.559	5.405	3.229	**1.360**	14.501
CMC	70.033	5.403	6.032	7.025	**3.278**	11.812	39.057
Sonar	99.635	4.938	5.573	6.459	4.189	**3.865**	11.678
Credit6000	70.528	33.297	35.857	52.305	**15.421**	217.895	391.099
Heart-statlog	59.316	4.770	5.267	6.453	**3.134**	4.455	11.919
Spamabse	247.696	19.128	23.238	22.507	**10.612**	91.235	445.082

5.5. Discussions

It is necessary to understand the drawbacks of each proposed stochastic algorithm. The algorithm proposed in this paper is designed to solve the specific optimization problem of feature selection, and the crossover operator redundancy is only for this class of optimization problems; the method may not be effective when solving continuous optimization problems. Secondly, the FS problem is optimized for a dataset, and the number of samples and the dimensionality of the data features are the most critical factors affecting the running time of the algorithm, so how to apply the algorithm to high-dimensional instances without increasing the computational complexity is a significant task for future research.

6. Conclusions

By summarizing the characteristics of the asexual genetic algorithm and analyzing the feature selection problem, we use the asexual genetic algorithm for the first time to solve the feature selection problem. The article demonstrates for the first time the advantages of the asexual genetic algorithm for solving this type of problem from a theoretical point of view, followed by the development and exploration ability of the algorithm by changing the population size and mutation rate to balance the algorithm to get the best variation rate for solving this type of problem, and finally the algorithm is used to solve the actual problem of Android Malware Application Detection, the results demonstrate that the algorithm is a feasible alternative in solving the feature selection problem.

Author Contributions: Conceptualization, L.W. and Y.G.; methodology, L.W.; formal analysis, S.G.; review and editing, X.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the National Natural Science Foundation of China under Grant (11961001), the Construction Project of first-class subjects in Ningxia higher Education (NXYLXK2017B09), the major proprietary funded project of North Minzu University (ZDZX201901), and postgraduate Innovation Project Funding of Northern University for Nationalities (YCX20087).

Institutional Review Board Statement: Article does not involve human research.

Informed Consent Statement: Article does not involve human research.

Data Availability Statement: The study did not report any data.

Acknowledgments: In this section you can acknowledge any support given which is not covered by the author contribution or funding sections. This may include administrative and technical support, or donations in kind (e.g., materials used for experiments).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Jain, A.K. Statistical pattern recognition: A review. *IEEE Trans. Pattern Anal. Mach. Intell.* **2002**, *27*, 4–37. [CrossRef]
2. Jesus, J.; Canuto, A.; Araujo, D. An exploratory analysis of data noisy scenarios in a Pareto-front based dynamic feature selection method. *Appl. Soft Comput.* **2020**, *100*, 106951. [CrossRef]
3. Nguyen, B.H.; Xue, B.; Zhang, M. A survey on swarm intelligence approaches to feature selection in data mining. *Swarm Evol. Comput.* **2020**, *54*, 100663. [CrossRef]

4. Ray, P.; Reddy, S.S.; Banerjee, T. Various dimension reduction techniques for high dimensional data analysis: A review. *Artif. Intell. Rev.* **2021**, *54*, 3473–3515. [CrossRef]
5. Rodrigues, D.; Pereira, L.A.M.; Almeida, T.N.S.; Papa, J.P.; Yang, X.S. BCS: A Binary Cuckoo Search algorithm for feature selection. *Proc. IEEE Int. Symp. Circuits Syst.* **2013**. [CrossRef]
6. Battiti, R. Using mutual information for selecting features in supervised neural net learning. *Neural Netw. IEEE Trans.* **1994**. [CrossRef] [PubMed]
7. Razniewski, S.; Strzelecki, M. Evaluation of texture features based on mutual information. *Ispa Int. Symp. Image Signal Process. Anal.* **2005**. [CrossRef]
8. Fleuret, F. Fast Binary Feature Selection with Conditional Mutual Information. *J. Mach. Learn. Res.* **2004**, *5*, 1531–1555.
9. Dash, M.; Liu, H. Feature selection for classification. *Intell. Data Anal.* **1997**, *1*, 131–156. [CrossRef]
10. Holland, J. Adaptation in natural and artificial systems: An introductory analysis with application to biology. *Control Artif. Intell.* **1975**. [CrossRef]
11. Cantó, J.; Curiel, S.; Martínez-Gómez, E. A simple algorithm for optimization and model fitting: AGA (asexual genetic algorithm). *Astron Astrophys.* **2009**, *501*, 1259–1268. [CrossRef]
12. Farasat, A.; Menhaj, M.B.; Mansouri, T.; Moghadam, M.R. ARO: A new model-free optimization algorithm inspired from asexual reproduction. *Appl. Soft Comput.* **2010**, *10*, 1284–1292. [CrossRef]
13. Simoes, A.; Costa, E. Using genetic algorithms with sexual or asexual transposition: a comparative study. *Proc. CEC00* **2000**, *10*, 1196–1203.
14. Amirghasemi, M.; Zamani, R. An effective asexual genetic algorithm for solving the job shop scheduling problem. *Comput. Ind. Eng.* **2015**, *83*, 123–138. [CrossRef]
15. Salesi, S.; Cosma, G. A novel extended binary cuckoo search algorithm for feature selection. In Proceedings of the International Conference on Knowledge Engineering and Applications, London, UK, 21–23 October 2017; pp. 6–12.
16. Ab Razak, M.F.; Anuar, N.B.; Othman, F.; Firdaus, A.; Afifi, F.; Salleh, R. Bio-inspired for Features Optimization and Malware Detection. *Arab. J. Sci. Eng.* **2018**, *43*, 6963–6979. [CrossRef]
17. Zhang, Y.; Wang, S.; Phillips, P.; Ji, G. Binary PSO with mutation operator for feature selection using decision tree applied to spam detection. *Knowl. Based Syst.* **2014**, *64*, 22–31. [CrossRef]
18. Palanisamy, S.; Kanmani, S. Artificial Bee Colony Approach for Optimizing Feature Selection. *IJCSI* **2012**, *9*, 432–438.
19. Sreedharan, N.P.; Ganesan, B.; Raveendran, R.; Sarala, P.; Dennis, B. Grey Wolf Optimization-based Feature Selection and Classification for Facial Emotion Recognition. *IET Biom.* **2018**, *7*. [CrossRef]
20. Hu, P.; Pan, J.S.; Chu, S.C. Improved Binary Grey Wolf Optimizer and Its application for feature selection. *Knowl. Based Syst.* **2020**, *195*, 105746. [CrossRef]
21. Pan, J.S.; Tian, A.Q.; Chu, S.C.; Li, J.B. Improved binary pigeon-inspired optimization and its application for feature selection. *Appl. Intell.* **2021**. [CrossRef]
22. Wang, J.W.; Chen, C.H.; Pan, J.S. Genetic Feature Selection for Texture Classification Using 2-D Non-Separable Wavelet Bases. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **1998**, *E81A*, 1635–1644.
23. Emary, E.; Zawbaa, H.M.; Hassanien, A.E. Binary grey wolf optimization approaches for feature selection. *Neurocomputing* **2016**, *172*, 371–381. [CrossRef]
24. Dua, D.; Graff, C. *UCI Machine Learning Repository*; University of California, School of Information and Computer Science: Irvine, CA, USA, 2019. Available online: http://archive.ics.uci.edu/ml (accessed on 17 July 2020).
25. Tein, L.H.; Ramli, R. Recent advancements of nurse scheduling models and a potential path. In Proceedings of the ICMSA 2010, Grand Seasons Hotel, Kuala Lumpur, Malaysia, 3–4 November 2010; pp. 395–409.
26. Feizollah, A.; Nor, B.; Salleh, R.; Amalina, F. Comparative study of k-means and mini batch k-means clustering algorithms in android malware detection using network traffic analysis. In Proceedings of the ISBAST 2014, Kuala Lumpur, Malaysia, 26–27 August 2014. [CrossRef]
27. Nath, H.V.; Mehtre, B.M. Static Malware Analysis Using Machine Learning Methods. In Proceedings of the SNDS-2014, Trivandrum, India, 13–14 March 2014; pp. 440–450.
28. Xiaosong, Z.; Khan, R.U.; Kumar, J.; Ahad, I.; Kumar, R. Effective and Explainable Detection of Android Malware Based on Machine Learning Algorithms. In Proceedings of the ICCAI 2018, Chengdu, China, 12–14 March 2018; pp. 35–40.
29. Zhao, X.; Fang, J.; Wang, X. Android malware detection based on permissions. In Proceedings of the ICICT 2014, Nanjing, China, 2 October 2014. [CrossRef]
30. Aung, Z.; Zaw, W. Permission-Based Android Malware Detection. *IJSTR* **2013**, *2*, 228–234.
31. Wang, S.; Chen, Z.; Yan, Q.; Yang, B.; Peng, L.; Jia, Z. A mobile malware detection method using behavior features in network traffic. *J. Netw. Comput. Appl.* **2019**, *133*, 15–25. [CrossRef]
32. Aafer, Y.; Du, W.; Yin, H. DroidAPIMiner: Mining API-Level Features for Robust Malware Detection in Android. *Springer Int. Publ.* **2013**, *127*, 86–103.
33. Arp, D.; Spreitzenbarth, M.; Hubner, M.; Gascon, H.; Rieck, K.; Siemens, C.E. DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket. In Proceedings of the NDSS, San Diego, CA, USA, August 2014. [CrossRef]
34. Wang, W.; Gao, Z.; Zhao, M.; Li, Y.; Liu, J.; Zhang, X. DroidEnsemble: Detecting Android Malicious Applications with Ensemble of String and Structural Static Features. *IEEE Access* **2018**, *6*, 31798–31807. [CrossRef]

35. Raymer, M.L.; Punch, W.F.; Goodman, E.D.; Kuhn Leslie, A.; Jain, A.K. Dimensionality reduction using genetic algorithms. *IEEE Trans. Evol. Comput.* **2000**, *4*, 164–171. [CrossRef]
36. Bhattacharya, A.; Goswami, R.T.; Mukherjee, K. A feature selection technique based on rough set and improvised PSO algorithm (PSORS-FS) for permission based detection of Android malwares. *Int. J. Mach. Learn. Cybern.* **2018**, *10*, 1893–1907. [CrossRef]
37. *Apktool*. May 2015. [Online]. Available online: https://ibotpeaches.github.io/Apktool/ (accessed on 17 July 2020).
38. Taheri, L.; Kadir, A.F.; Lashkari, A.H. Extensible Android Malware Detection and Family Classification Using Network-Flows and API-Calls. In Proceedings of the ICCST 2019, Cairo, Egypt, 1–3 October 2019. [CrossRef]
39. Kennedy, J.; Eberhart, R.C. A discrete binary version of the particle swarm algorithm. In Proceedings of the 1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation, Orlando, FL, USA, 12–15 October 1997. [CrossRef]
40. Saremi, S.; Mirjalili, S.; Lewis, A. *How Important Is a Transfer Function in Discrete Heuristic Algorithms*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 625–640.
41. Hilda, G.T.; Rajalaxmi, R.R. Effective feature selection for supervised learning using genetic algorithm. In Proceedings of the ICECS, Coimbatore, India, 26–27 February 2015; pp. 909–914.
42. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
43. Mirjalili, S.; Lewis, A. The Whale Optimization Algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
44. Colorni, A. Distributed optimization by ant colonies. *Proc. ECAL* **1991**, *142*, 134–142.
45. Hussien, A.G.; Hassanien, A.E.; Houssein, E.H.; Bhattacharyya, S.; Amin, M. S-shaped Binary Whale Optimization Algorithm for Feature Selection. *Recent Trends Signal Image Process.* **2019**, *727*, 79–87.
46. Wan, Y.; Wang, M.; Ye, Z.; Lai, X. A Feature Selection Method Based on Modified Binary Coded Ant Colony Optimization Algorithm. *Appl. Soft Comput.* **2016**, *49*, 248–258. [CrossRef]