A Quantum Feature Selection Method for Network Intrusion Detection

Mingze Li[1,2], Hongliang Zhang[3], Lei Fan[2], and Zhu Han[1]
Dept. of Electrical and Computer Engineering[1], Engineering Technology[2], University of Houston, TX, USA
Dept. of Electrical and Computer Engineering[3], Princeton University, NJ, USA

2022 IEEE 19th International Conference on Mobile Ad Hoc and Smart Systems (MASS) | 978-1-6654-7180-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/MASS56207.2022.00048

***Abstract*—Feature selection (FS) approaches rank features based on the score of the coefficients with labels. However, these selected features usually lead to a suboptimal solution to classification problems as they are selected independently. Moreover, with large dimensional feature sets, the computational complexity of FS algorithms can be prohibitively high. In this paper, we first formulate the feature selection problem as an integer programming model. Then we propose to utilize the strong computation ability of quantum annealers to solve the discrete integer programming problems, where the quantum annealer has the potential to be significantly faster than classical solvers to solve discrete optimization problems. We also design a wrapper algorithm to choose the optimal parameters of QUBO. Experiments show that our proposed strategy can select the representative features in the NSL-KDD dataset. Compared with HHO, WOA, PSO, and other algorithms, our strategy retains the least features to minimize the detection time, while the accuracy increase to 89.2%. Our algorithm also shows a good performance in computation time, detection rate and precision.**

***Index Terms*—Quantum annealing, quantum machine learning, feature selection, QUBO**

I. Introduction

In many machine learning problems, the data is composed of high dimensional feature vectors, which typically lead to performance degradation due to the curse of dimensionality [1]. Moreover, high-dimensional data contains noisy and redundant features, which will cause overfitting and make the data less interpretable. Feature selection (FS) is an effective solution to address these issues. To be specific, FS algorithms pick a relevant and possibly small portion of features for the classifier to be used in supervised learning. As a result, the algorithms' learning time is significantly reduced, and also the model's interpretability is improved.

In the literature, the FS algorithms could be classified by the way feature sets are evaluated and employed in data analysis, which defines filter, wrapper, and embedded techniques [2]. As a pre-processing phase before the learning algorithm, filter-based approaches rate the features and then pick those with high ranking scores. Wrapper-based approaches use the learning algorithm that will be used in the end to score the features. The learning algorithm is combined with FS in embedded approaches. Filter-based approaches for supervised FS are the focus of our research.

Generally, a filter-based FS problem can be formulated as a binary linear programming model, which maximizes a certain performance criterion. The performance criteria can be Pearson coefficients [3], mutual information (MI) [4] or others. One popular heuristic solution is to assign a score to each feature separately, and then pick the top-k ranked features. However, the features chosen by the heuristic methods are typically unsatisfactory for the following reasons. On one hand, the heuristic method calculates each feature's score separately, without taking the feature correlation into consideration. For example, it is possible that features a and b both have low scores, but the combination ab has a very high score [5]. The filter method does not consider features a and b in this scenario, even if both should be chosen. On the other hand, this method is not able to deal with redundant characteristics. For example, features a and b both have high scores, but they are significantly connected. Thus, both a and b will be chosen and lead to an increase of computation complexity. Another category of current researches treats the FS process as a global optimization problem in which a subset of characteristics is selected at the same time, which can better capture the correlations among features. The minimum test collection problem is such an example [6], in which binary variables are connected with features and constraints are set to quantify the coefficients between elements. However, it is difficult for classical solvers to get the solution for large-scale discrete optimization problems.

To overcome the issues mentioned above, we first build an integer programming model for FS and then reformulate this model as a QUBO model which can be solved by quantum annealers. Generally, the development of quantum computing techniques can be categorized into two directions: gate-based quantum computers and quantum annealers. Currently, the gate-based quantum computer is limited to less than 100 quantum bits [7]. D-Wave Systems Inc., on the other hand, manufactures quantum annealers with 5,000 quantum bits (qubits). The D-Wave quantum annealer has just been verified for its strong computation ability to solve binary optimization issues [8]. We use a general framework to reformulate the integer programming FS model into a QUBO model. In this framework, we divided features into multiple classes and select a certain number of features. To fully utilize the computation ability of quantum annealers, we also take the correlation between features and model them as the constraints in the problem. Then we use a wrapper method to determine the best combination of parameters for models, including the number of features selected from each class, etc. At last, we benchmark our proposed algorithm on NSL-KDD dataset to demonstrate the effectiveness of our method.

2155-6814/22/$31.00 ©2022 IEEE
DOI 10.1109/MASS56207.2022.00048

The paper is organized as follows. In Section II, we discuss the integer programming FS model and the way to reformulate it as a QUBO model. In Section III and IV, we show the case studies and experiments results. We conclude our work in Section V.

II. Optimization Models and Algorithms

In this paper, we aim to select $M(M \leq N)$ sub-features from N features, which make the classification result as close to the ground-truth distribution as possible. To achieve this, we formulate the FS problem as a binary linear programming model to maximize the relevant features while minimizing the redundant features. Then, we reformulate the binary linear model to a QUBO problem and use quantum annealers to solve it.

A. Integer Programming Model of FS

We assume our task is on a data set $\mathcal{D} := (\boldsymbol{f}_n^m, \mathbf{Y})$ with n elements and m features per data. Let $\mathbf{S} = \{1, 2, ..., m\}$ be the feature labels, and $\mathbf{S}_1, \mathbf{S}_2, ...\mathbf{S}_p$ represent p feature classes, where $\mathbf{S} = \mathbf{S}_1 \cap \mathbf{S}_2 \cap ... \cap \mathbf{S}_p$. We divide these features into multiple classes as a limited number of qubits are provided. We prefer to choose features that are helpful in classifying samples and provide valuable information to the data labels. At the same time, we remove features that are similar to other features to not feed machine learning algorithms duplicate data.

Define $\mathbf{x}$ as a binary decision variable vector that determines which features are chosen. Feature i is selected if x_i is 1, and otherwise feature i is not selected. With the notations defined before, we aim to get the best subset by optimally choosing the features. In the optimization model, the objective function contains two components. The first component depicts the impact of characteristics on the marked class [9], which is called importance component and represented in a form that grows as additional terms are added: $\sum_{i=1} I(f_i, y)x_i$. The importance vector $I(f_i, y)$ represents the common information of the individual features $x_1, \ldots, x_n$ with class label y, and is therefore a measure for the importance of each feature. With the objective function, the importance component is maximized.

On the other hand, we depict the independence by the second component, which is defined as $\sum_i \sum_{j<i} I(f_i, f_j)x_i x_j$. This matrix $I(f_i, f_j)$ represents the common information among the individual features, and therefore measures their redundancy. The objective function optimizes the similarity between the selected features and the data labels while minimizing the similarity between the features themselves.

The relative weighting of independence and importance components is represented by a parameter α $(0 \leq \alpha \leq 1)$. We balance the two parts by adjusting α to study their influence on the classification performance.

In sum, we have the optimization model as follows:

$$\min_{\boldsymbol{x}} \left\{ \frac{1-\alpha}{k(k-1)/2} \sum_i \sum_{j<i} I(f_i, f_j)x_i x_j - \frac{\alpha}{k} \sum_i I(f_i, y)x_i \right\} \tag{1}$$

$$s.t. \sum_{i \in \mathbf{S}_j} x_i = K_j, \quad \forall\ \mathbf{S}_j \subseteq \mathbf{S}; \tag{2}$$

$$\sum_{i \in \mathbf{C}_g} x_i \leq 1, \quad \forall\ \mathbf{C}_g \in \mathbf{C}; \tag{3}$$

$$\sum_{i \in \mathbf{E}_g} x_i = T_g, \quad \forall\ \mathbf{E}_g \in \mathbf{E}; \tag{4}$$

$$x_i - x_j \leq 0, \quad \forall (i, j) \in \mathbf{M}. \tag{5}$$

For constraint (2), we assume that the data set has p main categories. Then, we have a list of constant integers K_1 to K_p, which decides how many features will be selected from each category. It is worth mentioning that the number K_1 to K_p are decided by experiments in Section 2.

For constraint (3), it is given based on the fact that some features are redundant. For example, in $\mathbf{C}_g$, features a and b both have high scores, but they contain the same information. To minimize the redundancy of the subset, At most one can be selected. We have set $\mathbf{C}$ to store all the conflict sets.

For constraint (4), some features are essential and have the priority to be selected. For example, in $\mathbf{E}$, We to need select T_1 features. We have set $\mathbf{E}$ to store all these essential sets.

For constraint (5), some features are highly correlated. For example, feature a has a low score when b is not in the subset, but ab has a high score. That implies that, feature a should not be selected when b is not selected. We have set $\mathbf{M}$ to store all these groups.

Although building the FS problem as a binary integer program problem is able to increase the accuracy and interpretability, problem (1) is still NP-hard [10]. According to [11], Gurobi solves integer linear programs using the branch-and-bound algorithm, in which it splits the search space into smaller branches and discards the branches whose lower bound is higher than the current solution. Since the search space is exponentially increasing with the size of binary linear problems (BLP), the computational complexity of this approach is exponential in the worst case. As a result, when the size of binary variables is large, or the constraints are too complex, it will be difficult or even impossible for the classical solver to get a global optimal result.

Specially, the work in [12] shows that the worst-case complexity is $\mathcal{O}(Mb^d)$, where M is the cost of expanding subproblems, b is the branching factor, and d is the search depth. As d depends on the size of the BLP instance, making it actually worst-case exponential time.

B. QUBO Model

Quantum annealing (QA) provides a new method to solve BLP problems. According to [13], the computation complexity of QA is expected to be $\mathcal{O}(e^{\sqrt{N}})$, where N is the instance size.

	Notation	Description
Descision variable	x_i	equals to 1, if feature i is in the subset; otherwise, equals to 0
Parameters	n	number of total features
	k	number of selected features
	K_i	number of selected features in subset $\mathbf{S}_i$
	α	bias of importance and independence parts
Set	$\mathbf{S}$	Feature labels. $S = 1, 2, ..., m$
	$\mathbf{S}_i$	Feature subsets. $\mathbf{S} = \mathbf{S}_1 \cap \mathbf{S}_2 \cap ... \cap \mathbf{S}_n$
	$\mathbf{I}$	Vector of feature coefficients
	$\mathbf{Y}$	Labels of elements
	$\mathbf{C}_g$	Vector of conflict features. $\mathbf{C}_g$ = {1,3,4,...}
	$\mathbf{C}$	Set of conflict vectors. $\mathbf{C}_g \in \mathbf{C}$
	$\mathbf{E}_g$	Vector of essential features. $\mathbf{E}_g$ = {1,3,4,...}
	$\mathbf{E}$	Set of essential vectors. $\mathbf{E}_g \in \mathbf{E}$
	$\mathbf{M}$	Set of supplement features. $\mathbf{E}$ = {{1,3},{4,5},...}

Some researches have been proposed to utilize QA to solve integer programming problems [14]. To solve this problem efficiently, we further reformulate problem (1) to a QUBO problem, which can be solved by D-Wave quantum annealers.

In recent years, Quantum annealer has been developed as a new and effective approach to solve a QUBO problem. Quantum annealer can be used to solve discrete optimization problems with specific structures because it tends to retain lowest energy state. If a problem can be expressed as energy states of a system, it can be fed to quantum annealers to solve. The Quantum Processing Unit (QPU) of a quantum annealer, which functions like the CPU in a classical computer, is made up of connected qubits that create a graph topology. This creates a physical system whose energy is measured by a function of its states called the Hamiltonian [15]. An arbitrary QUBO problem can be expressed by a Hamiltonian function:

$$f = \sum_{i=0}^{N-2} \sum_{j=i+1}^{N-1} J_{i,j} \sigma_i^Z \sigma_j^Z + \sum_{i=0}^{N-1} h_i \sigma_i^Z, \tag{6}$$

where $\hat{\sigma}_i$ denotes a Pauli - z matrix acting on qubit i with eigenvalues ± 1. Coefficient $J_{i,j}$ decides whether $\hat{\sigma}_i$ and $\hat{\sigma}_j$ is related, which corresponds to the second part (independence) of our model, while h_i corresponds to the independence part. In our optimization problem, we definite selected feature vector as x^*, which contains the variables described in the previous section. Therefore, each of these variables is assigned to a given variable index i in the QUBO model. As $x_i \in \{0, 1\}$ and $\sigma_i \in \{+1, -1\}$. We reformulate x_i as $\left(1 - \sigma_i^Z\right)/2$. Thus, solving a QUBO problem is equivalent to finding a binary vector $\mathbf{x}^*$ which minimizes f

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \ f(\mathbf{x}). \tag{7}$$

We create a matrix $\mathbf{Q}$:

$$Q_{ii} = \alpha k * I(f_i, y), \quad i \in \mathbf{S}; \tag{8}$$

$$Q_{ij} = (1 - \alpha) I(f_i, f_j), \quad i \in \mathbf{S}, i \leq j, \tag{9}$$

where the diagonal matrix $\mathbf{Q}_{ii}$ corresponds to the importance part, and $\mathbf{Q}_{ij}$ corresponds to the independence part. Consequently, we can transform (1) to

$$\min_{\mathbf{x}} \ H_1 = \mathbf{x}^t \mathbf{Q} \mathbf{x}. \tag{10}$$

Algorithm 1 Reformulate and solve a QUBO model

Input: $\mathbf{I}$: Information matrix
$\mathbf{K}$: Set of number of features in each class
Output: features vector $\mathbf{x}^*$

1: Create the binary integer programming model (1) - (5).
2: Reformulate the objective function to QUBO problem.
3: Write constraints (2) - (5) to constraint satisfaction problems.
4: Add reformulated constraints (2) - (5) to QUBO.
5: Implement the QUBO model onto the quantum annealer.
6: **return** $\mathbf{x}^*$.

According to the definition, the QUBO model is unconstrained. However, constraints (2) - (5) need to be added to the Hamiltonian function added to QUBO model as a penalty term [16]. For example, constrain (2) is transformed to

$$H_2 = \sum_{i=1}^{z} P_1 \left(\sum_{i \in S_j} x_i - K_j \right)^2, \tag{11}$$

where P_1 is the penalty weight.

For constraint (3), the equivalent penalty is

$$H_3 = \sum_{q=1} P_2 x_i x_j, \tag{12}$$

If x_i and x_j are both equal to 1, the value of QUBO model will increase the penalty value. When one of them is 0, the value of QUBO model does not change. Thus, the model prefer to select at most 1 feature when the penalty is sufficiently high.

For constraint (4), the equivalent penalty is

$$H_4 = \sum_{r=1} P_3 \left(\sum_{i \in E_g} x_i - T \right)^2, \tag{13}$$

when T features are selected, the QUBO model gets the local minimum value.

Following the same transformation, we can transform constraint (5) to:

$$H_5 = \sum_{l=1} P_4 \left(x_i - x_i x_j \right)^2. \tag{14}$$

When x_i = 1 and x_j = 0, the value of QUBO model will increase, and otherwise it does not change. Thus, feature i could only be selected when j has been selected. We notice that all the reformulated constraints are polynomials of x_i and $x_i x_j$. Therefore, they are added to the matrix $\mathbf{Q}$, which ensure our final model is a QUBO problem. After the reformulation, we feed the formulated QUBO problem to the quantum annealer to solve. The whole proposed algorithm is presented in Algorithm 1.

C. Search the best $\mathbf{K}$

As mentioned before, in case that the number of qubits is not sufficient to solve the whole problem, we divide large feature sets into small subsets. Moreover, some data sets naturally

have multiple feature classes, while we only need to consider the independence defined in (1) within each class. However, a question arises that how to choose the optimal value of $\mathbf{K}$ for the QUBO model, which decides how many features are selected from each subset. Wrapper feature selection is a popular strategy for selecting the feature subset but takes a long time. This challenge can be alleviated by using a high-speed quantum solver to train parameters for our proposed Quantum feature selection (QFS) model.

At first, we solve the binary linear model without constraints (2). This means we use a flexible variable set $\mathbf{K}$ and decide the number of features in each class without limitations. Then we calculate the selected features in each class and use them as the initial set $\mathbf{K}$. Then we adjust the parameter set $\mathbf{K}$ by Algorithm 2 to search for the best combination that achieves the best performance. Thus, we transform the original objective of the selection from m features to adjusting $|\mathbf{K}|$ parameters, where $|\mathbf{K}| << m$. In each iteration, we maintain the smallest energy of the objective function to select the representative features. Obviously, the value initialization is very important. Our model is able to help decrease the possibility of dropping into a local minimum solution and use the least iterations to find the optimal solution during the value initialization. The convergence curve of our method is shown in Fig. 4. The ability of our method to find an optimal solution is at least not weak compared to other wrapper FS methods.

The whole proposed algorithm is presented in Algorithm 2. In the last paragraph, we already have an initial parameter set $\mathbf{K}$ and we use our QFS method to get an optimal feature set $\mathbf{x}$. $\mathbf{x}_i$ is the selected features in subsets $\mathbf{S}_i$, where $\mathbf{x} = \mathbf{x}_1 \cap \mathbf{x}_2 \cap ... \cap \mathbf{x}_p$. We perform the classification and obtain accuracy by using a classifier, such as the support vector classifier (SVC) [17]. If the current total feature number falls within the range of goal T, where T is the total number of selected features and $T = \sum_i K_i$, we select one subset from $\mathbf{S}$, such as S_1. Then, another feature will be chosen from this class. Assume K_1 features are chosen from S_1 in the last loop, we apply this feature subset to QA to select $K_1 + 1$ features. Assume $\mathbf{x}_1^*$ is the best outcome of S_1, we can use $\mathbf{x}_1^*$ to update $\mathbf{x}$. Let $\mathbf{x}^*$ to be $\mathbf{x}_1^* \cap \mathbf{x}_2 \cap ... \cap \mathbf{x}_p$, then use that value to determine a new accuracy. Each feature class will go through this method once again until the best result is found. The best accuracy can then be achieved by updating the set $\mathbf{K}$. If the greatest accuracy does not change, we end the loop.

We choose one class, and one less feature from that class may be chosen, if the current total number of features is more than the objective T. Continue like we did in the last sentence and give $\mathbf{K}$ an update. The set $\mathbf{K}$ can be utilized as the initial parameter set for target $T - 1$ after we obtain the best parameter set $\mathbf{K}$ of target T. Some optimal results of different T are shown in Table III.

Algorithm 2 Search the best $\mathbf{K}$

Input: T: Number of total selected features
$\mathbf{K}$: Set of initial parameters
$\mathbf{I}$: Information matrix.
α: weight balance parameter.
m: max iteration.
Output: The final set $\mathbf{K}^*$

$\mathbf{x}^* \leftarrow QFS(\mathbf{K}, \alpha, \mathbf{I})$;
$Max\_acc \leftarrow SVC(\mathbf{x}^*)$;
$\mathbf{K}^* \leftarrow \mathbf{K}$;
while iteration $< m$ **do**
  if sum($\mathbf{K}^*$) $\leq T$ **then**
    for $K_i \in \mathbf{K}^*$ **do**
      $K_i^* := K_i + 1$;
      $\mathbf{x}_i^* \leftarrow QFS(K_i^*, \alpha, \mathbf{I})$;
      Update $\mathbf{x}^*$ by $\mathbf{x}_i^*$;
      accuracy $a \leftarrow SVC(\mathbf{x}^*)$;
      Insert a to vector $\mathbf{A}$.
    if $Max(\mathbf{A}) > Max\_acc$ **then**
      $Max\_acc \leftarrow Max(\mathbf{A})$;
      Update $\mathbf{K}^*$.
  if sum($\mathbf{K}^*$) $> T$ **then**
    for $K_i \in \mathbf{K}^*$ **do**
      $K_i^* := K_i - 1$;
      $\mathbf{x}_i^* \leftarrow QFS(K_i^*, \alpha, \mathbf{I})$;
      Update $\mathbf{x}^*$ by $\mathbf{x}_i^*$;
      accuracy $a \leftarrow SVC(\mathbf{x}^*)$;
      Insert a to vector $\mathbf{A}$.
    if $Max(\mathbf{A}) > Max\_acc$ **then**
      $Max\_acc \leftarrow Max(\mathbf{A})$;
      Update $\mathbf{K}^*$.
return $\mathbf{K}^*$.

III. Case Study

In section II, we presented our novel QFS algorithm. The current section contains a study of different experiments to evaluate the performance of QFS. For this purpose, we use NSL-KDD dataset, which is synthetic and taken from real world data sources.

A. Data set

To verify the performance of the QFS algorithm in Intrusion Detection System (IDS) feature selection, we adopt the NSL-KDD network intrusion detection dataset for experiments.

1) Dataset Analysis: NSL-KDD data set is an improved version of KDDCUP99 [18], which is a DARPA98 IDS-based attack simulator that simulates four main types of attacks. Compared with the KDDCUP99 dataset, the NSL-KDD has all of the same characteristics as the original KDDCUP99, but it has been cleaned of redundant records and has had the proportion of connection types adjusted to make it more reasonable and dependable for classification tests. The NSL-KDD dataset is a classic dataset that has been used in the field of anomaly detection. It acts as a helpful benchmark for academics to contrast their proposed strategies against. KDDTraint+ and KDDTest+ are subsets of the NSL-KDD dataset. They are divided as the training set and the test set. The network attacks contain four types [19]:

TABLE I
THE FEATURES OF NSL-KDD DATASET.

Classification of features	Number	Name of features
The basic features of network connections	9	(1) duration, (2) protocol_type, (3) service, (4) flag, (5) src_bytes, (6) dst_bytes, (7) land, (8) wrong_fragment, (9) urgent
The content related features of network connections	13	(10) hot, (11) num_failed_logins, (12) logged_in, (13) num_compromised, (14) root_shell, (15) num_root, (16) su_attempted, (17) num_file_creations, (18) num_shells, (19) num_access_files, (20) num_outbound_cmds, (21) is_host_login, (22) is_guest_login
The time related traffic features of network connections	9	(23) count, (24) srv_count, (25) serror_rate, (26) srv_serror_rate, (27) rerror_rate, (28) srv_rerror_rate, (29) same_srv_rate, (30) diff_srv_rate, (31) srv_diff_host_rate
Host based traffic features of network connections	10	(32) dst_host_count, (33) dst_host_srv_count, (34) dst_host_same_srv_rate, (35) dst_host_diff_srv_rate, (36) dst_host_same_src_port_rate, (37) dst_host_srv_diff_host_rate, (38) dst_host_serror_rate, (39) dst_host_srv_serror_rate, (40) dst_host_rerror_rate, (41) dst_host_srv_rerror_rate

TABLE II
THE DISTRIBUTION OF SAMPLE CATEGORIES.

Data category	KDDTrain+	KDDTest+	Number of samples
Normal	65120	11536	76656
DoS	36944	6251	43195
Probe	10786	2421	13207
R2L	995	2653	3648
U2R	52	67	119
All	113897	22928	136825

- Denial of Service Attacks (DoS): It is an attempt by the attacker to restrict network usage by disrupting service availability to the intended users.
- User to Root Attacks (U2R): Occur when the attacker has access from a normal user account and tries to gain root access through system vulnerabilities.
- Remote to Local Attacks (R2L): The attacker does not have an account on local system but tries to gain access through sending network packets to exploit the vulnerabilities and gain access as a local user.
- Probing Attacks: Occur when the attacker scans the system network to collect information about the system in aims to use it for avoiding the system security control.

The detailed feature names and distribution of sample categories are shown in Tables I and II from [20]. The NSL-KDD dataset includes four types of features, which are the basic features of network connections (9 in total), the content related features of network connections (13 in total), the time related traffic features of network connections (9 in total), and the host based traffic features of network connections (10 in total), where each feature may be:

- an integer, for example, the number of times the system sensitive files and directories were accessed.
- a category, such as agreement type.
- a decimal number, representing, for example, percentage of REJ bad Connections.
- a boolean (0/1) value, which usually be considered as a category.

The data preprocessing is helpful to improve the accuracy of classification and ensure the reliability of the results. As the dataset has mixed data types of numbers and characters which are difficult to deal with, the one-hot encoding method is used

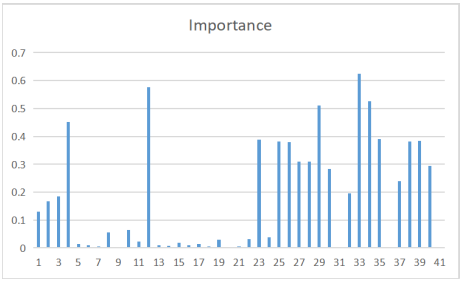

Fig. 1. Importance of each feature

to map different characters to different values. We convert the categorical variables to integer variables by assigning each category a class number. For the output label, we have classes (0 - 4), where class 0 represents the normal access, and class 1-4 represents each type of attack. Moreover, the dataset is normalized to eliminate the influence of features of different orders of magnitude on the calculation results, thus reducing the classification error. After one-hot encoding, the values of each feature are normalized to the interval [0, 1], and they can be normalized by as [20]:

$$x^* = \frac{x - x_{min}}{x_{max} - x_{min}}. \tag{15}$$

where x^* is the normalized value, x is the original value, and x_{min} and x_{max} represent the maximum and minimum values of the same feature vector.

Fig. 1 shows the similarity of each feature with the output labels by their Pearson correlation coefficient (PCC), which will be defined below, with the classification variable in Fig. 1, for example. At the left-hand end, there are four pretty significant features, followed by a drop. The coefficients with the label are higher for characteristics 23 to 40. As a result, the issue is figuring out how to pick up sufficient critical traits.

B. Basic Settings

There are several ways to evaluate the common information of two vectors. In this part, we introduce two common used functions to calculate the coefficient matrix $\mathbf{I}$, which contains the similarity between the feature vector and the label vector by $\mathbf{I}_{ii}$, and the similarity between the two features by $\mathbf{I}_{ij}$. The

Algorithm 3 Search for α. Given the information matrix $\mathbf{I}$, find the α that maximize the accuracy of classification method $SVC(\mathbf{x})$

Input: $\mathbf{I}$: Common information matrix.
s: step size.
Output: α: weight balance parameter.

```
α ← 0.05;
m_acc ← 0;
acc ← SVC(α, I);
while α < 1 do
    if m_acc < acc then
        m_acc ← acc;
        α ← α;
    α ← α + s;
    acc ← SVC(α, I);
return α.
```

matrix $\mathbf{I}$ can be transformed to the matrix $\mathbf{Q}$ in QUBO model by the method in section 2. As all the coefficients could be calculated in parallel, the run time of the algorithm matches the $\mathcal{O}(1)$ complexity. The functions to calculate matrix are listed below:

1) Pearson: The PCC is used to assess the linear correlation between two variables X, Y and can be determined using the provided formulation. The covariance of X and Y is represented by the function $\mathrm{Cov}(X, Y)$. The value of y might be anywhere between $+1$ and -1. $y = +1$ indicates that X and Y are totally positively correlated. $y = 0$ implies that X is not correlated to Y at all. Finally, a value of -1 indicates that X and Y are fully negatively correlated.

$$p = \frac{\mathrm{Cov}\,(X, Y)}{\sqrt{D\,(X)}\,\sqrt{D\,(Y)}}. \tag{16}$$

2) Mutual information: MI is a measure of how much information one random variable has about another variable, according to [21]. The MI is officially defined as follows:

$$I(x; y) = \sum_{i=1}^{n}\sum_{j=1}^{n} p(x(i), y(j)) \log\left(\frac{p(x(i), y(j))}{p(x(i))p(y(j))}\right). \tag{17}$$

where MI is zero when x and y are statistically independent.

As mentioned before, α is used as an optimization parameter, which defines the relative weighting of independence (greatest at α = 0) and importance (greatest at α = 1) [9]. After data preprocessing, we use a search method to determine the value of α which is most suitable for NSL-KDD dataset. With an initial α, we can create the $\mathbf{Q}$ by α and $\mathbf{I}$ by (8) and (9). Then, with the QFS method (without constraint (2)), we get a optimal feature set $\mathbf{x}$. Apply $\mathbf{x}$ to a SVC classification method we get the accuracy acc. Thus, we definite a function $acc = SVC(\alpha, \mathbf{I})$. The detailed search process is shown in Algorithm 3.

Fig. 2 shows the results by tuning α from 0.1 to 0.9. The feature independence component has a higher weighting on the left side when α is near to zero, while the model emphasising the feature importance when α is near to one. We notice that when α = 0.75 the model get the best accuracy.

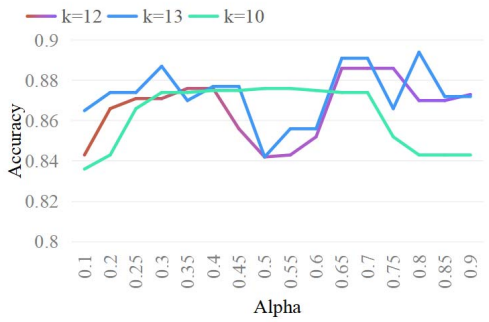

Fig. 2. QUBO feature selection with an increasing α

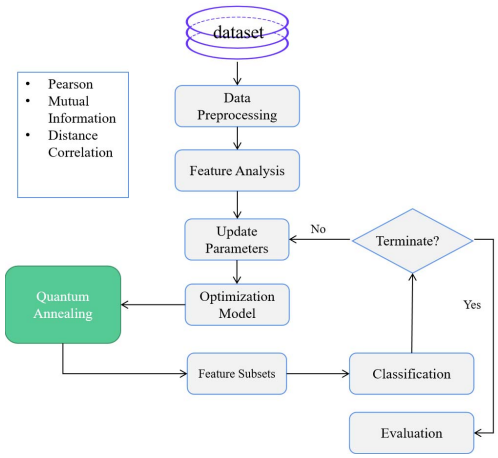

Fig. 3. Flow chart of the training process

C. Testing Process

In Fig. 3, we depict how the experimental system was intended to function in terms of feature selection and classification techniques. Following the data preprocessing, we compute the correlations between features and labels using the previous two metrics, including Pearson and MI. The data will be utilized to determine the parameters of the QFS Model. After that, we get the optimal feature subsets by QFS and compare the performance with other FS models.

IV. Experiments Result

In this section, we compare our method with other feature selection methods by evaluating the accuracy of various classification models trained on the respective feature subsets.

Classification performance is an important evaluation criterion of feature selection for intrusion detection. Several classification performance metrics can be found in the review of [22].

- TP—True Positive: The label data is positive and the prediction result is positive.

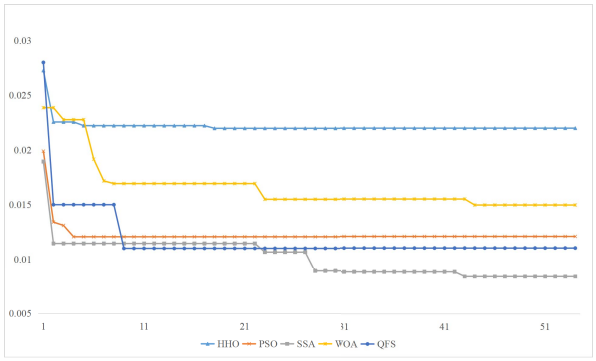

Fig. 4. Convergence curve of different methods

TABLE III
SAMPLES OF SELECTED FEATURE SETS

Number of features	Selected set of features
9	[1,2,3,28,32,33,34,35,38]
12	[0,1,2,3,7,28,32,33,34,35,36,40]
14	[0,1,2,3,4,5,7,28,32,33,34,35,37,38]
19	[0,1,2,3,4,5,6,7,16,20,22,32,33,34,35,36,37,38,39]

TABLE IV
FEATURE SELECTION RESULTS OF DIFFERENT FEATURE SELECTION ALGORITHMS

	Amount	Features	ACC
QFS	12	[0,1,2,3,7,28,32,33,34,36,38,40]	0.894
HHO	18	[0,1,2,3,4,5,9,11,18,20,23,28,29,30,35,36,37,38]	0.862
SSA	21	[1,4,5,8,10,11,15,17,18,19,20,21,26,27,28,29,33,35,36,37,39]	0.856
WOA	14	[0,3,4,5,7,8,10,12,17,24,27,28,35,39]	0.878
PSO	13	[0,2,4,5,7,15,18,19,25,28,29,34,40]	0.882

- FN—False Positive: The label data is positive and the prediction result is negative.
- FP—False Positive: The label data is negative and the prediction result is positive.
- TN—True Negative: The label data is negative and the prediction result is negative.

There are several metrics that may be used to assess feature selection algorithms including:

- Accuracy: Accuracy is the percentage of accurately predicted samples to all samples that were forecasted, which ranges from (0, 1). Accuracy measures the classifier's overall performance and is calculated as follows:

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN}. \tag{18}$$

- Precision: Precision is the capability of a classification model to identify only the relevant data points in the dataset. It is estimated by the ratio of correctly predicted positive samples to all predicted positive samples as follows:

$$precision = \frac{TP}{TP + FP}. \tag{19}$$

- Detection rate: Detection rate is also called recall. It estimates the ability of a model to discover all the relevant data points by the ratio of the number of correctly predicted positive samples to the total number of true positive samples. It is estimated as follows:

$$recall = \frac{TP}{TP + FN}. \tag{20}$$

SVC models are then trained using the specified feature subsets. We do cross validation on each model and present the mean and standard deviation of classification accuracy.

In this paper, we compare the performance of different number of selected features. The results of different feature selection algorithms are shown in Table IV. According to Table IV, QFS algorithm selects 12 most representative features from the NSL-KDD dataset, accounting for 29.26% of the total number of features. Compared with HHO [23], SSA [24], WOA [25] and PSO [26] algorithms, the total number of features in the four types of attack datasets is reduced by 14.6%, 21.9%, 4.8%, 2.4%.

Then we measured the classification performance using the selection results of HHO, SSA, WOA, PSO, and QFS algorithms as feature subsets. SVC classifier is used as the classification algorithm. The classification accuracy of different algorithms is shown in Table IV and V. Our QFS algorithm achieved around 90% accuracy, improved by 3.2%, 4.8%, 1.6% and 1.2% compared to the other four FS algorithms. Fig. 5, 6, 7 show that the accuracy, precision and detection rate of feature subset produced by different feature selection algorithms. For Normal, Probe, DoS, and R2L datasets, the average precision of QFS is 89.8%. It increases by 2.70%, 8.30%, 2.88%, and 6.34% compared to HHO, SSA, WOA and PSO algorithms. The performance of R2L is not as good as the other types because the samples of R2L only account a small part of training and testing sets. It would be a better way to train a separate feature subset for R2L attacks.

We also measure the performance of the two evaluation method mentioned in Section III in Table VI. We found that Pearson coefficients retains more information and achieves better performance. Compared with MI, the weighted average precision, recall and accuracy rate of four types increases by 3.7%, 5.4% and 3.2%.

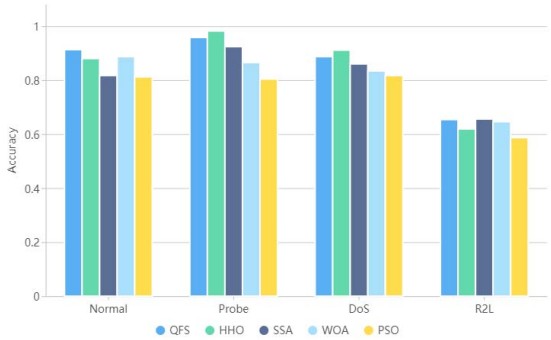

Fig. 5. Classification accuracy on different algorithms

Fig. 6. Classification precision on different algorithms

Fig. 7. Classification recall on different algorithms

TABLE V
COMPARISON BETWEEN SEVERAL FEATURE SELECTION ALGORITHMS USING NSL-KDD DATASET

	Accuarcy				
Data Category	QFS	HHO	SSA	WOA	PSO
Normal	0.852	0.846	0.817	0.887	0.812
Probe	0.985	0.937	0.924	0.865	0.804
DoS	0.803	0.887	0.860	0.834	0.817
R2L	0.987	0.919	0.856	0.846	0.787

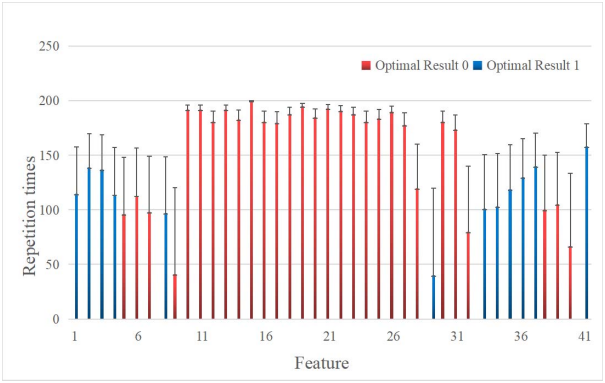

Fig. 8. Occurrence of features of the samples

It seems QA is much better than classical solvers. However, the performance of D-wave annealers is still limited by the noise. With the increase of problem size, more qubits are needed, which also leads to the increase of noise. Thus, the

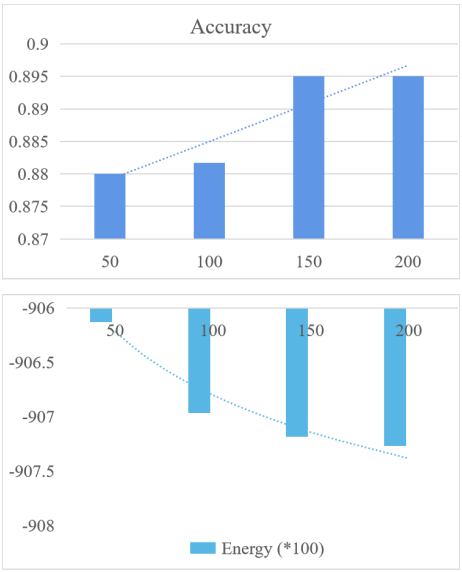

Fig. 9. Repetition of samples from the D-Wave quantum annealer

actual computation complexity depends on the complexity of problems. To overcome the randomness of the D-wave annealer, 200 same samples are performed to find the best optimal result. In Fig. 8, the occurrence and deviation of features in the 200 repetitions are shown. The blue bar indicates that this feature's label is set to 1 and that it was selected in the optimal outcome, while the red bar indicates that it was not selected. Fig. 9 shows with the repetition of QFS, the lowest energy decreases and the accuracy increases.

Fig. 10. Feature selection and detection time of different algorithms

In Table VII, we compare the feature selection and detection time of five alternative algorithms to further highlight the benefit of our QFS technique. The time spent removing irrelevant and redundant features is referred to as feature selection time. The detection time is an indication of how long the SVC classifier took to train the model with the representative feature subsets we chose. It can be seen from Fig. 10 that the feature selection time of QFS algorithm is shorter than other algorithms, which indicates that QFS algorithm can find the optimal solution with a higher speed while maintaining the accuracy. At the same time, QFS algorithm remove redundant features, which considerably increases the detection speed. In comparison to the other four feature selection algorithms, the detection time of QFS algorithm is reduced by 64.83%,

TABLE VI
COMPARISON BETWEEN TWO DIFFERENT INFORMATION EVALUATION METHODS

Data Category	Mutual Information			Pearson			Amount of Elements
	precision	recall	accuracy	precision	recall	accuracy	
Normal	0.836	0.978	0.901	0.864	0.969	0.913	9711
Dos	0.987	0.909	0.946	0.982	0.935	0.958	5740
Probe	0.826	0.740	0.781	0.805	0.986	0.887	1106
R2L	0.994	0.424	0.783	0.995	0.494	0.754	2199

TABLE VII
COMPUTATION TIME OF DIFFERENT ALGORITHMS

	QFS	HHO	SSA	WOA	PSO
Detection time (s)	24.02	70.48	56.68	28.22	26.72
Selection time (s)	786.87	1689	2579.91	3042.12	2793

50.91%, 7.94%, and 6.12% .

V. CONCLUSION

In this study, we proposed an integer programming model for FS that balances the feature subset's importance and independence. We then considered the interaction between features and reformulated it into a QUBO problem, which is different from past work. Our experiments have been run on quantum hardware, which further demonstrates that our algorithm is effective. To demonstrate our framework's effectiveness, we performed experiments by comparing different common feature selection methods and the resulting performance on different classifiers. Our experimental results showed that our method outperforms other methods.

ACKNOWLEDGEMENT

This work is partial supported by U.S. NSF CNS-2107216, CNS-2128368, and EPCN-2045978.

REFERENCES

[1] F. Daum and J. Huang, "Curse of dimensionality and particle filters," in *IEEE Aerospace Conference Proceedings (Cat. No.03TH8652)*, vol. 4, Big Sky, MT, May. 2003.

[2] A. Jović, K. Brkić, and N. Bogunović, "A review of feature selection methods with applications," in *38th International Convention on Information and Communication Technology, Electronics and Microelectronics*, Opatija, Croatia, May. 2015, pp. 1200–1205.

[3] J. Benesty, J. Chen, Y. Huang, and I. Cohen, *Pearson Correlation Coefficient*. Berlin, Heidelberg: Springer Berlin Heidelberg, Mar. 2009, pp. 1–4. [Online]. Available: https://doi.org/10.1007/978-3-642-00296-0_5

[4] P. A. Estevez, M. Tesmer, C. A. Perez, and J. M. Zurada, "Normalized mutual information feature selection," *IEEE Transactions on Neural Networks*, vol. 20, no. 2, pp. 189–201, Jan. 2009.

[5] Q. Gu, Z. Li, and J. Han, "Generalized fisher score for feature selection," in *27th Conference on Uncertainty in Artificial Intelligence*, Barcelona, Spain, Jul. 2011.

[6] P. Bertolazzi, G. Felici, P. Festa, G. Fiscon, and E. Weitschek, "Integer programming models for feature selection: New extensions and a randomized solution algorithm," *European Journal of Operational Research*, vol. 250, no. 2, pp. 389–399, Apr. 2016.

[7] E. National Academies of Sciences, Medicine *et al.*, *Quantum computing: progress and prospects*. Washington, DC: National Academies Press, 2019.

[8] R. Li, Y. He, S. Zhang, J. Qin, and J. Wang, "Cell membrane-based nanoparticles: a new biomimetic platform for tumor diagnosis and treatment," *Acta Pharmaceutica Sinica B*, vol. 8, no. 1, pp. 14–22, Jan. 2018.

[9] A. Milne, M. Rounds, and P. Goddard, "Optimal feature selection in credit scoring and classification using a quantum annealer," *White Paper 1Qbit*, Apr. 2017. [Online]. Available: http://1qbit.com/files/white-papers/1QBit-White-Paper-—Optimal-Feature-Selection-in-Credit-Scoring-and-Classification-Using-a-Quantum-Annealer_-_2017.04.13.pdf

[10] P. M. Pardalos and S. Jha, "Complexity of uniqueness and local search in quadratic 0–1 programming," *Operations Research Letters*, vol. 11, no. 2, pp. 119–123, Mar. 1992.

[11] Gurobi. Mixed-Integer Programming (MIP) – A Primer on the Basics. [Online]. Available: https://www.gurobi.com/resource/mip-basics/

[12] D. R. Morrison, S. H. Jacobson, J. J. Sauppe, and E. C. Sewell, "Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning," *Discrete Optimization*, vol. 19, pp. 79–102, Feb. 2016.

[13] S. Mukherjee and B. K. Chakrabarti, "Multivariable optimization: Quantum annealing and computation," *The European Physical Journal Special Topics*, vol. 224, no. 1, pp. 17–24, Feb. 2015.

[14] Z. Zhao, L. Fan, and Z. Han, "Hybrid quantum bendersx2019; decomposition for mixed-integer linear programming," in *2022 IEEE Wireless Communications and Networking Conference (WCNC)*, Austin, TX, Apr. 2022, pp. 2536–2540.

[15] R. Nembrini, M. Ferrari Dacrema, and P. Cremonesi, "Feature selection for recommender systems with quantum computing," *Entropy*, vol. 23, no. 8, p. 970, Jul. 2021.

[16] F. Glover, G. Kochenberger, and Y. Du, "A tutorial on formulating and using qubo models," *arXiv preprint arXiv:1811.11538*, 2018.

[17] N. Cristianini, J. Shawe-Taylor *et al.*, *An introduction to support vector machines and other kernel-based learning methods*. Cambridge, UK: Cambridge university press, 2000.

[18] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *IEEE Symposium on Computational Intelligence for Security and Defense Applications*, Ottawa, Canada, Jul. 2009.

[19] H. Alazzam, A. Sharieh, and K. E. Sabri, "A feature selection algorithm for intrusion detection system based on pigeon inspired optimizer," *Expert Systems with Applications*, vol. 148, p. 113249, Jun. 2020.

[20] L. Xin, Y. Peng, W. Wei, J. Yiming, and T. Le, "Lnnls-kh: a feature selection method for network intrusion detection," *Security and Communication Networks*, vol. 2021, Jan. 2021.

[21] P. Isola, D. Zoran, D. Krishnan, and E. H. Adelson, "Crisp boundary detection using pointwise mutual information," in *Computer Vision – ECCV 2014*. Cham, Switzerland: Springer International Publishing, Sep. 2014.

[22] R. Padilla, S. L. Netto, and E. A. B. da Silva, "A survey on performance metrics for object-detection algorithms," in *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, Niteroi, Brazil, 2020.

[23] J. Too, A. R. Abdullah, and N. Mohd Saad, "A new quadratic binary harris hawk optimization for feature selection," *Electronics*, vol. 8, no. 10, 2019.

[24] M. Seyedali, A. H. Gandomi, Z. M. Seyedeh, S. Shahrzad, F. Hossam, and M. M. Seyed, "Salp swarm algorithm: A bio-inspired optimizer for engineering design problems," *Advances in Engineering Software*, vol. 114, pp. 163–191, Dec. 2017.

[25] M. Sharawi, H. M. Zawbaa, E. Emary, H. M. Zawbaa, and E. Emary, "Feature selection approach based on whale optimization algorithm," in *2017 Ninth International Conference on Advanced Computational Intelligence (ICACI)*, Doha, Feb. 2017, pp. 163–168.

[26] S. B. Sakri, N. B. Abdul Rashid, and Z. Muhammad Zain, "Particle swarm optimization feature selection for breast cancer recurrence prediction," *IEEE Access*, vol. 6, pp. 29 637–29 647, Jun. 2018.