Parallel Intrusion Detection Method Based on ReliefF and Improved Crow Search Optimization Postprint

Authors: Ma Chao

Date: 2018-08-13T00:00:00+00:00

Abstract

The increase in network data volume leads to increased computational and time complexity. To improve the detection accuracy and speed of network intrusion detection, a novel intrusion detection method designated as RICSA-KELM is proposed. First, the ReliefF filtering technique is employed to eliminate irrelevant features and noise, thereby reducing feature dimensionality. Subsequently, a wrapper-based approach utilizing an Improved Crow Search Algorithm (ICSA) is implemented for optimal feature subset selection, concurrently achieving parameter optimization for the Kernel Extreme Learning Machine (KELM) classifier. The designed linear weighted objective function maximizes classification accuracy while simultaneously minimizing the false alarm rate and the number of selected features. Furthermore, a multi-threaded parallel computing method based on a multi-core platform is introduced to further optimize the model's computational approach and improve computational efficiency. The performance of RICSA-KELM is empirically evaluated and analyzed using the KDD99 and UNSW-NB15 benchmark datasets. Experimental results demonstrate that the proposed model outperforms SVM, ELM, KNN, and other comparative methods, exhibiting high detection accuracy, rapid detection efficiency, and low false alarm rate, thereby establishing itself as an effective network intrusion detection method.

Full Text

A Parallel Intrusion Detection Method Based on ReliefF and Improved Crow Search Optimization

College of Digital Media, Shenzhen Institute of Information Technology, Shenzhen, Guangdong 518172, China

Abstract

The increasing volume of network data has led to higher computational and time complexity. To improve the detection accuracy and speed of network intrusion detection, this paper proposes a novel intrusion detection method called RICSA-KELM. First, the ReliefF filter method is employed to remove irrelevant features and noise, thereby reducing feature dimensionality. Then, a wrapper method based on an improved crow search algorithm (ICSA) is used for optimal feature subset selection while simultaneously optimizing the parameters of the kernel extreme learning machine (KELM) classifier. The designed linear-weighted objective function considers maximum classification accuracy while minimizing the false alarm rate and feature subset size. Additionally, a multi-threaded parallel computing method based on a multi-core platform is proposed to further optimize the model' s computational approach and improve efficiency. Experiments using the KDD99 and UNSW-NB15 benchmark datasets were conducted to test and analyze the performance of RICSA-KELM. The results demonstrate that the proposed model outperforms SVM, ELM, KNN, and other methods, achieving high detection accuracy, fast detection efficiency, and low false alarm rates, making it an effective network intrusion detection method.

Keywords: crow search algorithm; intrusion detection; parallel computing; kernel extreme learning machine; ReliefF

0 Introduction

With the growing popularity of Internet technology, network attack methods have become increasingly diversified, and both the number and severity of attacks continue to rise. Traditional security measures such as firewalls and data encryption can no longer meet modern cybersecurity requirements. As an active defense technology, network intrusion detection can collect and analyze system security data from networks, extract various behavior patterns and characteristics, and provide real-time security protection with timely alerts, making it a research hotspot in the information security field.

Intrusion detection typically includes misuse detection and anomaly detection. The former achieves high detection rates but cannot identify novel attack variants, while the latter can discover new intrusion behaviors. Consequently, current research primarily focuses on anomaly-based intrusion detection. In pattern recognition, network intrusion detection is essentially a multi-classification problem comprising two main components: feature selection and classifier design. In recent years, with the rapid development of artificial intelligence technology, an increasing number of researchers have attempted to apply AI techniques to network intrusion detection. For instance, in 2016, Bamakan et al. improved the time-varying chaotic particle swarm optimization (PSO) algorithm and combined it with multi-criteria linear programming (MCLP) and support vector machine (SVM) for intrusion detection research, validating the method' s effectiveness on the KDD dataset. In the same year, Li Cong et al. proposed a

network intrusion detection algorithm called FAST-ABQGSA-SVM that integrates FAST feature selection with an adaptive binary quantum gravitational search algorithm. In 2017, Hua Huiyou et al. proposed a Cluster-KNN algorithm that fuses K-means and KNN for network intrusion detection, combining clustering and classification for offline preprocessing and online classification phases, demonstrating advantages in accuracy, false alarm rate, and miss rate compared to other methods in the field. Also in 2017, Akashdeep et al. proposed an intrusion detection system based on information gain (IG) for feature ranking combined with an artificial neural network (ANN) classifier, achieving excellent results on the KDD-99 dataset. Wang et al. proposed an intrusion detection framework based on enhanced features and SVM, transforming original features into a higher-quality feature space through density ratio, resulting in more robust performance on the KDD dataset. Raman et al. proposed using a hypergraph genetic algorithm (GA) to simultaneously optimize SVM parameters and feature selection for intrusion detection, obtaining good results on the KDD dataset. Aburomman et al. designed a weighted SVM multiclass classifier based on differential evolution for intrusion detection, achieving excellent classification results on the NSL-KDD dataset. In 2018, Hajisalem et al. proposed a hybrid classification method combining artificial bee colony (ABC) and artificial fish swarm (AFS) for intrusion detection research, achieving high accuracy and low false alarm rates on both KDD and UNSW-NB15 datasets. Chiba et al. proposed an optimized BPNN method for anomaly intrusion detection research, also achieving good results.

From these studies, we can observe that network intrusion detection models and methods based on artificial intelligence technology have received widespread attention. Among them, SVM and neural network-based models are most commonly used and have achieved good results. However, four main problems remain:

a) For neural network methods, weight parameter optimization typically relies on gradient descent-based approaches, which easily fall into local minima, require long training times, and need multiple iterations to converge, resulting in slow learning speeds. For SVM models, the selection of kernel functions and parameters has a significant and direct impact on intrusion detection results, yet there are no unified standards or theoretical guidelines for parameter selection. Most researchers currently use swarm intelligence algorithms for iterative parameter optimization, but the search process is prone to falling into local optima and failing to find the global optimum.

b) Feature selection and classifier optimization are equally important but are usually performed independently, with previous work rarely considering the correlation between them.

c) Although SVM methods can handle binary classification problems effectively, they cannot be directly applied to multi-class problems like intrusion detection and require complex "one-vs-one" or "one-vs-many" con-

structions.

d) Most existing methods are based on serial execution, resulting in low computational efficiency and failing to consider further improvements in network intrusion detection efficiency under parallel conditions.

Based on the above analysis, to overcome these shortcomings and further improve detection accuracy and efficiency—while considering that feature selection and classifier optimization are correlated and have equally important impacts on detection precision—we propose the RICSA-KELM model for network intrusion detection. The model employs a hybrid feature selection method combining filter and wrapper approaches. First, ReliefF is used for feature dimensionality reduction to remove irrelevant features and noise. Then, an improved crow search algorithm (ICSA) is proposed to simultaneously achieve optimal feature subset selection and classifier parameter optimization. Continuous and discrete ICSA algorithms are integrated into the model, with the continuous version automatically adjusting KELM parameters and the discrete version selecting optimal feature subsets. Chaotic operations are introduced to increase population diversity and better balance global and local search. OpenMP-based multi-threaded parallel methods on multi-core platforms are used to improve ICSA algorithm performance, fully utilizing CPU resources to enhance algorithm performance and further improve computational efficiency.

1 ReliefF Algorithm

The ReliefF algorithm, proposed by Konomenko et al. as an extension of the Relief algorithm, is a filter-based feature selection method applicable to multi-class problems. It evaluates feature relevance and redundancy by calculating nearest neighbor samples within the same class and across different classes. The algorithm randomly selects a sample subset p from the original dataset, then chooses s nearest neighbor samples from the same class as p and s nearest neighbor samples from different classes, computing and updating feature weight values. This process repeats until the relevance between features and class labels is obtained for all samples. Features are then sorted in descending order according to their weight values, and a subset is selected based on a given threshold—features with weights above the threshold form the new feature subset, while those below are removed.

The ReliefF algorithm implementation is as follows:

Input: p sample instances and their corresponding feature attributes.

Output: Feature weight vector $\mathbf{w}$.

a) Initialize $\mathbf{w} = 0$.

b) Select one sample from p and choose s nearest neighbor samples from both the same class and different classes, then calculate feature weight values. The weight calculation formula is:

$$w_i = w_i - \frac{\sum_{j=1}^{s} \text{diff}(I, P_j, H_j)}{m \cdot s} + \frac{\sum_{C \neq \text{class}(P_i)} \left[\frac{p(C)}{1-p(\text{class}(P_i))} \sum_{j=1}^{s} \text{diff}(I, P_j, M_j(C)) \right]}{m \cdot s}$$

where m is the number of sample samplings; the diff function calculates the distance between two sample instances regarding feature I; $M_j(C)$ is the j-th nearest neighbor sample from a different class; and class(p_i) is the class label of sample p_i.

ReliefF offers advantages including easy extensibility, strong effectiveness, good stability, and high computational efficiency. It can quickly process large amounts of data and noisy data, making it an excellent filtering evaluation algorithm. Since the feature evaluation process in ReliefF considers correlations between features, it can effectively remove irrelevant features.

2 Improved Crow Search Algorithm (ICSA)

The Crow Search Algorithm (CSA), proposed by Askarzadeh in 2016, is a novel swarm intelligence optimization algorithm that simulates the intelligent foraging behavior of crows in nature. CSA is simple to implement, robust, and involves few parameters that require tuning, making it suitable for network optimization and other application domains. Crows are highly intelligent social birds that typically hide surplus food after finding it; the hiding location is called a memory value. When needed, they can retrieve the food and can also track other crows to steal their food. The tracked crows can protect their food with a certain awareness probability (AP).

2.1 Continuous CSA Algorithm

When solving optimization problems, assume N crows are randomly distributed in an n-dimensional search space. $x_{i,t} = [x_{i,t}^1, x_{i,t}^2, \ldots, x_{i,t}^n]$ ($i = 1, 2, \ldots, N$; $t = 1, 2, \ldots,$ Maxiter) represents the position of the i-th crow at iteration t. $M_{i,t}$ denotes the memory value of crow i at iteration t, representing its optimal position. $AP_{i,t}$ represents the awareness probability of crow i at iteration t, and $fl_{i,t}$ denotes its flight length. The algorithm initializes control parameters including population size M, awareness probability AP, flight length fl, and maximum iterations Maxiter.

Traditional CSA randomly initializes positions using the formula:

$$x_{i,t} = \text{rand} \cdot (x_{\max} - x_{\min}) + x_{\min}$$

where $x_{i,t}$ is the randomly generated position of a crow; $x_{\max}$ and $x_{\min}$ are the maximum and minimum values of x; and rand is a random number in [0,1].

However, random initialization cannot guarantee individual quality, as some solutions may be far from the optimal position. Good initial populations facilitate solution efficiency and quality, while poor initialization affects efficiency and increases uncertainty. An effective initialization ensures faster algorithm convergence. This paper addresses this issue by using chaotic mapping for crow position initialization. Chaotic motion is a seemingly random behavior that naturally emerges in deterministic nonlinear systems, possessing both deterministic and stochastic properties. This nonlinear system characteristic enables algorithms to escape local optima while searching for global optimal solutions. Therefore, we adopt the Logistic chaotic mapping function for crow position initialization:

$$X_{n+1} = \mu \cdot X_n \cdot (1 - X_n), \quad \mu \in [0, 4], X_n \in (0, 1)$$

where parameter μ controls the degree of chaos.

During iteration t, crow i randomly selects another crow j to track and steal food from. The algorithm includes both global and local search components, dynamically balanced through the awareness probability AP. When a randomly generated number is greater than or equal to crow j' s awareness probability AP, crow j detects crow i and leads it to a random position. Conversely, when the random number is less than AP, crow j remains unaware and crow i moves toward crow j' s optimal position. Since position updates affect both the optimal solution and convergence speed, we introduce chaotic algorithms to further optimize the crow search position update. The position update expression is:

$$x_{i,t+1} = \begin{cases} x_{i,t} + w_i \cdot fl_{i,t} \cdot (m_{j,t} - x_{i,t}) & \text{if } r_i \geq AP_{i,t} \\ x_{i,t} + w_z \cdot fl_{i,t} \cdot (x_{i,t} - x_{j,t}) & \text{if } r_i < AP_{i,t} \end{cases}$$

where w_i represents the chaotic mapping value at generation i; w_z represents the chaotic mapping value at generation z; $AP_{i,t}$ is crow j' s awareness probability at generation t; and r_i and r_j are uniformly distributed random numbers in [0,1].

Equation (4) shows that the hybrid function further balances global and local search through more flexible dynamic perturbation. In early stages, larger w_i values ensure global search dominates, improving population diversity. In later stages, smaller w_i values increase local search weight, accelerating convergence.

When crow i' s position changes, the memory value is updated as follows:

$$M_{i,t+1} = \begin{cases} x_{i,t+1} & \text{if } f(x_{i,t+1}) > f(M_{i,t}) \\ M_{i,t} & \text{otherwise} \end{cases}$$

where $M_{i,t}$ represents the crow's memory value and $f(M_{i,t})$ represents its fitness value.

2.2 Discrete CSA Algorithm

To more effectively handle practical problems, Sayed et al. proposed a discrete CSA algorithm for feature selection. Each dimension of the population individual and the optimal position is either 0 or 1, introducing a mapping function $S(x)$ to convert continuous space values to discrete space [0,1]:

$$S(x) = \frac{1}{1 + e^{-x}}$$

[Figure 1: see original paper] illustrates the overall flow of the ICSA algorithm.

3 RICSA-KELM Model

This chapter details the RICSA-KELM classification model. The model first uses the ReliefF filtering method to remove irrelevant features and noise. Then, under a parallel environment, it adaptively determines KELM parameters and identifies the most discriminative feature subset, employing the ICSA algorithm to simultaneously perform parameter optimization and feature selection. In the proposed RICSA-KELM model, the fitness function is designed by considering three sub-objectives: the ACC value obtained by the KELM model, the false alarm rate, and the size of the feature subset. The overall model flow is shown in [Figure 2: see original paper]. Below, we describe the serial algorithm followed by the parallel implementation. Note that this stage assumes ReliefF-based filtering has already been applied to the original sample set for feature dimensionality reduction, removing irrelevant features and noise.

3.1 Serial Model

The RICSA-KELM model operates in three phases: Phase 1 uses the ICSA algorithm to iteratively search for the optimal feature set and KELM parameter combination; Phase 2 trains the RICSA-KELM classifier on different training datasets using the optimal feature set and parameters from Phase 1; Phase 3 tests the trained classifier on the test dataset. The linear-weighted multi-objective function simultaneously considers classification accuracy (ACC), false alarm rate, and feature count. The main steps are:

a) Encode solutions with length n+2 dimensions, where the first n dimensions consist of binary values (1 indicates the feature is selected, 0 indicates it is not), and the last two dimensions represent the continuous values of KELM parameters C and γ. The solution encoding format is $X = [0, 1, \ldots, 1, 0, C, \gamma]$.

b) Initialize the population and set relevant parameters including population size, maximum iterations, awareness probability AP, and flight length *fl*.

c) Train on KELM using the feature set and parameters decoded from the initialized individuals.

d) Higher ACC, lower false alarm rate, and smaller feature subset yield higher fitness values. Therefore, a linear-weighted multi-objective function is designed as follows:

$$f_1 = \frac{1}{K}\sum_{i=1}^{K} \text{accuracy}_i = \text{ACC}$$

$$f_2 = 1 - \text{FA}$$

$$f_3 = \frac{1}{n}\sum_{j=1}^{n}(1 - m_j)$$

$$F = \mu \cdot f_1 + \eta \cdot f_2 + \sigma \cdot f_3$$

where f_1 represents the ACC value obtained from K-fold cross-validation; f_2 incorporates FA (false alarm rate); f_3 involves feature values where m is the feature count and n is the total number of features. In F, μ, η, and σ are constants representing the weights for KELM accuracy, false alarm rate, and selected feature subset, respectively, with $\mu + \eta + \sigma = 1$. These weights can be adjusted to appropriate values depending on each sub-objective's contribution to the evaluation result. Since classification performance depends more on accuracy and false alarm rate, experimental trials set μ, η, and σ to 0.5, 0.3, and 0.2, respectively.

e) Increment iteration count: $t = t + 1$.

f) Update population individuals' positions and memory values using equations (4) and (5).

g) Use the updated solutions from step f) to decode feature sets and parameters, train on KELM, and calculate individual fitness values using equations (7)-(10).

h) Record the current population' s best solution. If the current fitness value exceeds the stored best fitness value, update it; otherwise, retain the stored value.

i) If the maximum population size is reached, proceed to step j); otherwise, return to step f).

j) Compare the current fitness value with the global best fitness value. If the current value is greater, update the global best; otherwise, retain the historical best.

k) If the maximum iteration count is reached, proceed to step l); otherwise, return to step e) to continue iterative optimization.

l) Output the global optimal solution and decode it to obtain the optimal feature subset and parameter combination (C, γ).

m) Train the optimal classifier model on KELM using the optimal feature subset and parameter combination with the training set.

n) Test on the test set and obtain final classification results.

3.2 Parallel Model

For many complex optimization problems, the ICSA algorithm requires numerous updates to guarantee finding the optimal solution. The initial solution generation, fitness calculation, and memory value update in ICSA are time-consuming but independent operations, giving the algorithm natural parallelism. To fully exploit this parallelism and improve efficiency, we propose implementing the parallel model using OpenMP on multi-core processors. The multi-core platform framework consists of three layers:

a) ICSA-KELM Model Layer: Composed of a series of population individuals, the parallel algorithm controls the entire CSA iteration process, with each individual participating independently in the computation.

b) OpenMP Platform Layer: Ensures synchronization of the parallel algorithm and establishes communication with the operating system. The core component is a scheduler that provides job scheduling and allocation to the OS.

c) Multi-core Processor Layer: Jobs are system-called through OpenMP at this layer.

The pseudo-code for parallel RICSA-KELM is as follows:

```
Initialize model parameters
Train KELM
Calculate the fitness
while i < max_{iteration}
    for each particle
        Update position
        Update memory
        Train KELM
        Calculate the fitness
        Calculate fitness_{best}
        Calculate memory_{best}
    end for
    Calculate fitness_{global}
    Calculate memory_{global}
    i = i + 1
end while
```

4 Experiments

4.1 Data Description and Processing

The experiments utilize the KDD99 and UNSW-NB15 datasets, which are widely used in network intrusion detection research. KDD99 is the most classic dataset, with many intrusion detection methods benchmarked against it. UNSW-NB15 was created in 2015 by the Australian Centre for Cyber Security (ACCS) research group. Dataset information is shown in and . Due to space limitations, detailed feature descriptions for UNSW-NB15 are not provided.

To reduce differences in feature magnitudes, data normalization is applied to preprocess the datasets, mapping all feature values to [0,1] to avoid interference from large-scale data on small-scale data and ensure result validity:

$$x_i = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}$$

where $x_{\min}$ and $x_{\max}$ are the minimum and maximum values in the dataset. Additionally, to avoid overfitting and underfitting and ensure convincing results, a double cross-validation method is employed: inner 10-fold cross-validation determines the optimal feature subset and parameters, while outer 5-fold cross-validation evaluates KELM classification performance. Since a single cross-validation run cannot guarantee fairness due to random data partitioning, multiple runs are conducted.

4.2 Experimental Settings

The proposed RICSA-KELM algorithm was implemented in MATLAB 2014b using the ELM toolbox. The hardware platform features an Intel quad-core processor at 3.2 GHz, 16 GB RAM, and 64-bit Windows 8. For fair comparison, ICSA-KELM model settings match those of PSO-SVM. The search ranges for KELM and SVM parameters C and γ are $C \in \{2^{-10}, \ldots, 2^{15}\}$ and $\gamma \in \{2^{-15}, \ldots, 2^{5}\}$.

Detailed parameter settings for RICSA-KELM model training and algorithm comparison are as follows: initial particle positions and velocities are random numbers in [0,1], population size is 20, and maximum iterations is 100.

4.3 Experimental Results and Analysis

To validate the proposed model's effectiveness, we first compare the classification performance of RICSA-KELM with three other models (KELM, SVM, LSSVM) on both datasets, as shown in and . The RICSA-KELM model uses ICSA for parameter optimization and feature selection, while SVM and LSSVM use grid search for parameter optimization. The tables present accuracy (ACC) and false alarm rate (FA) for each dataset, where higher ACC and lower FA indicate better performance.

On the KDD99 dataset, the proposed model achieves the highest average classification accuracy of 95.88%, significantly outperforming the other three methods. The ranking follows as KELM, LSSVM, and SVM. The proposed model improves classification accuracy by 1.04%, 3.64%, and 3.46% compared to the other three models, respectively, while achieving the lowest false alarm rate of 1.28%. Similarly strong results are obtained on the UNSW-NB15 dataset.

and compare RICSA-KELM performance with and without feature selection on both datasets. On KDD99, the feature-selected model improves ACC by 1.72% compared to the non-selected version; on UNSW-NB15, the improvement is 1.83%. This demonstrates the excellent performance of RICSA-KELM, attributed to the improved ICSA algorithm' s ability to automatically adjust and optimize parameters for optimal classification performance.

To statistically validate the classification performance improvement, Wilcoxon signed-ranks tests are conducted with a 95% confidence interval. The results show significant differences ($P < 0.05$) between RICSA-KELM and the non-feature-selected model across all four performance metrics on both datasets, indicating substantial performance improvements. Additionally, the proposed model exhibits relatively small variance, demonstrating good stability.

For comprehensive comparison, we implement PSO-optimized KELM (PSO-KELM) and GA-optimized KELM (GA-KELM). As shown in , the proposed model achieves higher ACC and lower FA than both PSO-KELM and GA-KELM, reflecting ICSA' s stronger search and optimization capability compared to PSO and GA.

To investigate the feature selection process, lists the feature subsets selected by RICSA-KELM across 10-fold CV on KDD99. The dataset contains 41 features, but not all contribute to classification accuracy. Feature selection improves precision, as demonstrated in and . [Figure 3: see original paper] shows feature selection frequencies, with the most important features being F1, F2, F3, F4, F10, F20, F23, F24, F38, and F39 (selected $ $7 times). These correspond to duration, protocol_{type}, service, flag, hot, num_{{{outbound}}*{{cmds}}}, *count,* *srv*{count}, dst_{{{host}}*{{serror}}*}{rate}, and dst_{{{host}}*{{srv}}*}{{{serror}}_{{rate}}} in KDD99, providing valuable insights for network intrusion analysis.

To validate parallel model performance, compares parallel and serial models on KDD99. While performance metrics are similar (differences due to random cross-validation partitioning), the serial model' s average runtime is approximately three times longer than the parallel model. [Figure 4: see original paper] shows that the parallel model' s CPU time per fold is significantly lower, demonstrating that the proposed method benefits from parallelization, overcoming the time-consuming nature of traditional serial algorithms and greatly improving computational efficiency.

[Figure 5: see original paper] investigates ICSA' s global search capability and convergence speed by comparing optimal fitness value evolution between ICSA

and CSA on the first fold of KDD99. The ICSA curve shows rapid initial improvement, converging to its maximum by the 25th iteration, while CSA converges later (26th iteration) and reaches a lower final fitness value, sometimes falling into local optima. This confirms that ICSA possesses superior global search ability and faster convergence speed, efficiently synchronizing feature selection and parameter optimization in a parallel environment.

5 Conclusion

Intrusion detection is a critical and challenging research area in network information security. Efficient intrusion detection models with high prediction rates can better handle frequent and complex real-time network attacks. This paper proposes a parallel intrusion detection model called RICSA-KELM based on a hybrid filter-wrapper approach. The model uses ReliefF filtering for feature dimensionality reduction to eliminate irrelevant features and noise, then integrates continuous and discrete ICSA algorithms to simultaneously achieve optimal feature subset selection and classifier parameter optimization. This approach effectively removes redundant and irrelevant features from network data, reduces data dimensionality, and improves algorithm efficiency and classification capability. ICSA enhances CSA population diversity through chaotic operations while strengthening local search ability, improving global optimization capability and convergence speed. A linear-weighted multi-objective function comprehensively considers classification accuracy, false alarm rate, and feature count. OpenMP shared-memory parallelization achieves optimized classifier parallel computation, significantly reducing CPU runtime and improving algorithm efficiency.

Experimental results on KDD99 and UNSW-NB15 datasets demonstrate that compared with existing and similar methods, the proposed model obtains superior parameter combinations and feature sets, achieves significantly improved computational efficiency, and delivers better classification results. The model outperforms CSA-KELM, PSO-SVM, GA-KELM, and PSO-KELM, achieving higher detection accuracy, lower false alarm rates, and improved detection efficiency, making it an effective network intrusion detection model.

Future research directions include investigating other swarm intelligence algorithms such as ABC, and exploring ensemble methods combining multiple classifiers to further improve intrusion detection accuracy beyond the limitations of single classifiers.

References

[1] Muhammad F, Muhammad Sher, Yaxin Bi. Flow-based intrusion detection: techniques and challenges [J]. Computers & Security, 2017, 70: 238-254.

[2] Ashfaq R A R, Wang Xizhao, Huang J Z, et al. Fuzziness based semi-supervised learning approach for intrusion detection system [J]. Information

Sciences, 2017, 378: 484-497.

[3] Seyed M H B, Huadong W, Tian Y, et al. An effective intrusion detection framework based on MCLP//SVM optimized by time-varying chaos particle swarm optimization [J]. Neurocomputing, 2016, 199: 90-102.

[4] Li Cong, Yan Renwu, Zhu Changshui. Network intrusion detection based on FAST feature selection and ABQGSA-SVM [J]. Application Research of Computers, 2017, 34 (7): 2172-2179.

[5] Hua Huiyou, Chen Qimai, Liu Hai, et al. Hybrid K-means with KNN for network intrusion detection algorithm [J]. Computer Science, 2016, 43 (3): 158-162.

[6] Akashdeep, Ishfaq Neeraj feature reduced intrusion detection system using ANN classifier [J]. Expert Systems with Applications, 2017, 88: 249-257.

[7] Huiwen Wang, Shanshan Wang. An effective intrusion detection framework based on SVM with feature augmentation [J]. Knowledge-Based Systems, 2017, 136: 130-139.

[8] Raman M R G, Nivethitha S, Kannan K. An efficient intrusion detection system based on hypergraph-genetic algorithm for parameter optimization and feature selection in support vector machine [J]. Knowledge-Based Systems, 2017, 134: 1-12.

[9] Abdulla A A, Mamun B. A novel weighted support vector machines multi-class classifier based on different evolution for intrusion detection systems [J]. Information Sciences, 2017, 414: 225-246.

[10] Hajisalem V, Babaie S. A hybrid intrusion detection system based on ABC-AFS algorithm for misuse and anomaly detection [J]. Computer Networks, 2018, 136: 37-50.

[11] Chiba Z, Noureddine A, Khalid M. A novel architecture combined with optimal parameters for back propagation neural networks applied to anomaly network intrusion detection [J]. Computers & Security, 2018, 75: 49-66.

[12] Huang Y, McCullagh P J, Black N D. An optimization of ReliefF for classification in large datasets [J]. Data & Knowledge Engineering, 2009, 68 (11): 1348-1356.

[13] Askarzadeh A. A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm [J]. Computers & Structures, 2016, 169: 1-12.

[14] Wang G G, Guo L, Gandomi A H, et al. Chaotic krill herd algorithm [J]. Information Sciences, 2014, 274: 17–34.

[15] Sayed G I, Aboul E H, Ahmad T A. Feature selection via a novel chaotic crow search algorithm [J]. Neural Computing & Applications, 2017, 1: 1-13.

[16] Ma Chao, Ouyang J, Chen H L, et al. A novel kernel extreme learning machine algorithm based on self-adaptive artificial bee colony optimisation [J]. International Journal of Systems Science, 2016, 47 (6): 1348-1359.

[17] Jaroslav H, Michal L, Stanislav Z. Parallelization of interpolation, solar radiation and water flow simulation modules in GRASS GIS using OpenMP [J]. Computers & Geosciences, 2017, 107: 20-27.

[18] Moustafa N, Slay J. The evaluation of network anomaly detection systems: statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set [J]. Information Security Journal: A Global Perspective, 2016, 25: 1-3.

[19] Wang Mingjing, Chen Huiling, Yang Bo, et al. Toward an optimal kernel extreme learning machine using a chaotic moth-flame optimization strategy with applications in medical diagnoses [J]. Neurocomputing, 2017, 267 (6): 69-84.

[20] Ye Zhifan, Yu Yuanlong. Network intrusion classification based on extreme learning machine [C]// Proc of IEEE International Conference on Information and Automation. 2015: 1642-1647.

[21] Zhou Guangping, Shrestha A. Efficient intrusion detection scheme based on SVM [J]. Journal of Networks, 2013, 8 (9): 2128-2134.

[22] Zhang Hongmei, Gao Haihua, Wang Xingyu. Construct sparse least squares support vector machine for network intrusion detection [J]. Journal of East China University of Science and Technology, 34 (6): 876-881.

[23] Veček N, Črepinšek M, Mernik M. On the influence of the number of algorithms, problems, and independent runs in the comparison of evolutionary algorithms [J]. Applied Soft Computing, 2017, 54 (3): 23-45.

Note: Figure translations are in progress. See original paper for figures.