OPTIMIZED FEATURE SELECTION FOR INTRUSION DETECTION SYSTEM THROUGH META HEURISTIC BASED BPSO ALGORITHM

Selim Mahmud, Roll: 0424052079
Md. Ashraful Islam Bhuiyan, Roll: 0423052096
Navid Bin Mahamud, Roll: 0422052051

February 10, 2025

Contents

Chapter 1

Introduction

Intrusion detection systems (IDS) play a crucial role in cybersecurity because they can detect malicious activity and attempts to gain unauthorized access within a network. When a cyber-attacker tries to gain unauthorized access to a device, network, or system to breach security i.e. Confidentiality, Integrity, and Availability (CIA), it is called intrusion. An intrusion detection system (IDS) is a security tool that monitors computer networks or systems for any intrusion, i.e. malicious activity (cyber-attacks, any hacking attempts) or policy violations. IDS analyzes network traffic or system behavior and alerts administrators or users when it detects potentially harmful activities or any intrusion. This includes unauthorized access, malware outbreaks, or unusual patterns that may indicate a security breach. The intrusion detection system is a software tool or network device that can detect malicious traffic and take action against those malicious traffics. We will design an effective IDS system based on a meta heuristic approach named binary particle swarm optimization (BPSO) to increase detection capability within a reasonable time frame.

1.1 Background and Motivation

An intrusion detection system (IDS) monitors network traffic for suspicious activity and sends alerts when such activity is discovered. Anomaly detection and reporting are the primary functions of an IDS, but some systems also take action when malicious activity or abnormal traffic is detected. Such actions include blocking traffic sent from suspicious IP addresses. An IDS can be contrasted with an intrusion prevention system (IPS), which also monitors network packets for detecting unusual network traffic, much like an IDS. However, an IPS has the primary goal of preventing threats once detected instead of primarily detecting and recording threats. In modern network systems, there is a huge amount of network traffic data with a lot of different types of payload. This large dataset will obviously have a large number of features which makes it difficult to select good set of weighted features to classify the network payload whether the payload is normal or malicious. Meta heuristic optimization algorithms can be used to address this problem by iteratively searching for a set of weights that minimizes the classification error rate in classification decision. Here we want to use particle swarm optimization approach to address the problem by searching weights which minimizes error rate in payload feature selection and increases accuracy in intrusion detection in an acceptable time frame.

1.2 Objective

Our objective is to use Binary Particle Swarm Optimization (BPSO) to choose features effectively, to increase detection accuracy, to lower computing complexity and to ensure anomaly based detection. Our approach will give below improvements in Intrusion Detection System (IDS)

1. **To Optimize Feature Selection Using BPSO**
   - To find the most important and pertinent characteristics from high-dimensional datasets (like KDD Cup 99), create and implement a feature selection technique based on BPSO.
   - Evaluate the effect of feature subset selection on lowering computing expenses and enhancing detection precision.
2. **To Enhance Detection Accuracy and Efficiency**
   - Integrate machine learning models (such as Decision Tree, Random Forest, SVM, or Neural Networks) with BPSO-optimized features to get high detection accuracy for a variety of intrusion types.
   - Determines including precision, recall, F1-score, and computing time are used to compare the suggested method's performance with existing models.
3. Develop explanations and illustrations for the chosen attributes and detection outcomes to make sure that cybersecurity experts can understand and utilize the system.
4. Selection of optimal feature subset with essential information in an acceptable time with more accuracy.

1.3 Literature Review

Different type of Intrusion Detection Systems are there and they use different types of algorithms for detection mechanism. The exponential expansion of threats and the expansion of methods for attack via the communication channels of several recently launched Internet services only worsen the security dilemma. Numerous solutions have been put out to address the security concerns; nevertheless, the majority of current solutions fail to detect security threats effectively and with high performance.

Consequently, this study proposes a hybridization of modified binary Gray-Wolf Optimization with particle swarm Optimization. In this study, NSL KDD'99 and UNSW-NB15 are two benchmarking datasets that are used in the suggested solution and get better result [1]

1.4 Challenges

For intrusion detection, the KDD-CUP 99 dataset is commonly used. However, there are certain challenges in detecting intrusion.

1. Data volume is very high: In any enterprise network, volume of network packets is huge. To process this large volume data, large time and huge processing resource are necessary.

2. High Dimensionality of Data: Many characteristics, some of which are repetitive or unnecessary, are present in datasets such as KDD Cup 99, which are frequently utilized for intrusion detection research. As a result, detection performance is decreased and computational complexity rises.

3. Accuracy vs. Scalability: A couple of intrusion detection techniques are already in use. It is hard to balance between accurate detection and scalability to manage real-time scenarios or massive datasets.

4. Feature Selection Bottleneck: One crucial yet computationally costly operation is determining which features are most pertinent to maximize the detection process.

5. Changing Attack Patterns: Static or rules-based intrusion detection systems are less efficient due to the rapid evolution of security threats.

6. Model Generalization: Due to excessive fitting or dependence on particular attack patterns, many IDS models cannot be generalized to various network settings or datasets.

1.5 Possible Outcome

1. **Enhanced Intrusion Detection Accuracy**
   - In comparison to systems that use the whole feature set or conventional feature selection techniques, the suggested system will achieve improved detection accuracy through the use of Binary Particle Swarm Optimization (BPSO) to choose the most relevant characteristics.
   - Enhanced recall and precision for different types of attacks in data sets such as NSL-KDD or KDD Cup 99.
2. **Reduction in Computational Complexity**
   - The dimensions of the data set will be greatly reduced by the optimized feature subsets, which will be used to speed up the testing and training of the machine learning model.
   - The system will be better suited for real-time intrusion detection applications with increased efficiency.
3. **Scalable and Adaptive Framework**
   - The suggested system would indicate scalability by effectively managing large-scale datasets or real-time data streams without compromising performance.
   - Long-term performance will be ensured via dynamic feature selection or model updates that can adapt to changing attack patterns.

1.6 Experimental Design

A, B, C, and D are the phases involved in this research: investigation, design, development, and analysis of results. This study will examine how the existing method works for the intrusion detection system throughout the investigation phase. The aim of this phase is to find the shortcomings of existing intrusion detection techniques.

In the final phase, The KDD CUP 99 dataset will be used to examine the effectiveness and performance of the current approach. The following are the steps involved in the experimental design:

1. Data Preprocessing
2. Feature Selection Using BPSO

3. Model Training and Testing
4. Performance Evaluation
5. Result Analysis

Chapter 2

Methodology

An intrusion detection system (IDS) is a type of security tool used to identify malicious activity or unauthorized access to a computer system, network, or application. By limiting the risks connected with security breaches and offering early alerts about possible assaults, IDS plays a crucial part in cybersecurity.

For the purpose of tracking and evaluating user, process, and network traffic behavior in order to detect any unusual activity, intrusion detection systems are crucial. Security administrators might employ these technologies to identify vulnerabilities, safeguard critical data, and maintain business continuity.

2.1 Intrusion Detection Techniques

An intrusion detection system (IDS) monitors network traffic for potentially harmful transactions and promptly notifies users when it is detected. It is software that scans for harmful activity or rule breaches on a network or system. Several methods are used by IDS to identify intrusions:

2.1.1 Signature-based Detection

Signature-based detection relies on predefined patterns or signatures of known attacks. A system will trigger an alarm if it notices activity or traffic that matches a known signature or pattern. This technique is effective at detecting known attacks but struggles with identifying new or unknown threats. Incoming network traffic or system activity is scanned using signature-based detection, which compares the results to a database of recognized attack patterns, or ”signatures.” In basic terms, these signatures are distinct identifiers that correspond to harmful activities, such as certain code sequences, byte patterns, or actions characteristic of known threats.

2.1.2 Anomaly-Based Detection

Anomaly-based detection works by establishing a baseline of normal system or network behavior. Anomaly-based detection identifies deviations from normal system behavior. This technique does not rely on signatures but instead looks for unusual activity that might indicate an intrusion. It is more effective in detecting novel attacks but has higher false-positive rates than signature-based detection. Anomaly based IDS uses machine learning or statistical techniques to provide a baseline of what ”normal” behavior looks like in order to track network or system operations. Activity is marked as possibly harmful when it dramatically varies from this baseline. This approach depends on the idea that unusual behavior is probably dangerous.

2.2 Metaheuristic Algorithms

Modern computational methods such as metaheuristic algorithms are applied to address complicated optimization issues when more conventional approaches could be ineffective or fail to produce a solution. These algorithms, which are made to efficiently search vast solution areas, are frequently influenced by natural processes or occurrences. The details of meta-heuristic algorithms show the diversity of their mechanisms, adaptability, and potential applications, making them strong optimization tools. Every metaheuristic algorithm uses a particular trade-off between local search and randomization. It is essential to remember that the phrases ”heuristic” and ”metaheuristic” are not commonly defined in the literature, and some writers use them interchangeably. The genetic algorithm (GA) [2], particle swarm optimization (PSO) [3], binary particle swarm optimization (PSO) [4], ant colony optimization (ACO) [5], and artificial bee colony (ABC) [6] are the popular metaheuristic algorithms for optimization techniques. The GA, PSO, and BPSO optimization techniques are described in detail below.

2.2.1 Genetic Algorithm (GA)

A method of search and optimization that draws inspiration from natural selection is the Genetic Algorithm (GA). In order to enhance potential solutions to a given problem repeatedly, it imitates biological evolution concept as selection, crossover, mutation, and inheritance. GA was essentially founded on the ideas of natural selection in biological systems and Darwin’s hypothesis [2].

The following list outlines the GA's operational steps:

- Step 1: Initialize population P with N individuals randomly.
- Step 2: Evaluate fitness of each individual in P
- Step 3: While stopping criteria not met:
  - Select parents from P based on fitness
  - Perform crossover to create offspring
  - Apply mutation to offspring
  - Evaluate fitness of offspring
  - Replace some or all of P with offspring
- Step 4: Return the best individual as the solution.

Due to its efficiency, applicability, and simplicity, the genetic algorithm has been effectively used to solve various optimization problems in several academic sectors.

2.2.2 Particle Swarm Optimization (PSO)

The communal behavior of fish schools, bird flocks, and other swarming species served as the inspiration for the metaheuristic optimization technique known as particle swarm optimization (PSO) [3]. It is used to improve potential solutions iteratively in order to address optimization challenges.

A swarm consists of multiple particles, each representing a potential solution to the optimization problem. Each particle has a position, a velocity, its best position, knowledge of the best position found by the entire swarm. Finally, the swarm can be defined as follows:

$$\boldsymbol{S} = \{X_1, X_2, \dots X_N\} \tag{2.1}$$

where N denotes to particles (possible solutions) that can be defined as:

$$X_i = \{X_{i1}, X_{i2}, \dots X_{in}\},\ i = 1, 2, \dots N. \tag{2.2}$$

Particles are given arbitrary indices, and N is a user-defined algorithm parameter. The particles are supposed to move repeatedly in the search space, A. The objective function, *f(x)*, is assumed to be available for all points in A. Thus, each particle has a unique function value, f_i=$f(x_i) \in$ Y. This is accomplished by altering their position using a correct position shift, represented as:

$$\mathbf{V}_i = \{V_{i1}, V_{i2}, \dots V_{in}\},\ where\, i = 1, 2, \dots N. \tag{2.3}$$

In PSO, the velocity can be adjusted repeatedly to permit particles to visit the different regions of the search space A. If t is the current iteration, then $\mathbf{V}_i$ and $\mathbf{X}_i$ of the i-th particle will be represented as $\mathbf{X}_i^t$ and $\mathbf{V}_i^t$, respectively.

The velocity of the i-th particle is updated according to the information gained in previous steps. The task of the velocity updating of the i-th particle is performed by the use of memory, in which each particle stores the best position it has visited during the previous generations. PSO maintains a memory set in addition to $\boldsymbol{S}$, which stores the positions of the particles in the swarm. The new memory set $\boldsymbol{P}$ is as follows:

$$\boldsymbol{P} = \{P_1, P_2, \dots P_N\} \tag{2.4}$$

This $\boldsymbol{P}$ stores the best position of the i-th particle that visited in the previous generations:

$$P_i = \{P_{i1}, P_{i2}, \dots P_{in}\},\ i = 1, 2, \dots N. \tag{2.5}$$

The best position of the i-th can be defined as:

$$P_i^t = \arg\min_i f_i^t, \tag{2.6}$$

where t represents the iteration. Because PSO is developed based on the social behavior of bird flocks, the particles should share some sort of information to permit particles to share experiences between them. With the best position ever visited by all particles, the method approximates the global minimizer. Let G represent the best positions stored in $\boldsymbol{P}$ then the global best position (in case of minimization) can be determined as follows:

$$P_g^t = \arg\min_i f(P_i^t). \tag{2.7}$$

PSO is thus defined by the equations below.

$$V_{ij}^{t+1} = w \cdot V_{ij}^t + c_1 R_1^t (P_{ij}^t - X_{ij}^t) + c_2 R_2^t (P_g^t - X_{ij}^t), \tag{2.8}$$

$$X_{ij}^{t+1} = X_{ij}^t + V_{ij}^{t+1} \tag{2.9}$$

where $i = 1, 2, \dots N$ and $j = 1, 2, \dots n$, t is the current iteration, R_1 and R_2 are random variables with uniform distributions inside [0,1], while c_1, c_2 are weighting

Algorithm 1 Pseudocode for PSO.

```
1: **Initialize the swarm:**
2: Set the number of particles $N$
3: Set the number of particle Size $noV$
4: Initialize position $X_{ij}^t$ and velocity $V_{ij}^t$ for each particle randomly within a predefined range
5: Initialize personal best position $P_{ij}^t$ and global best position $P_g^t$
6: Set the maximum number of iterations $Max\_Iter$
7: Set the parameters: inertia weight $w$, cognitive coefficient $c_1$, and social coefficient $c_2$
8: **Evaluate the fitness of each particle:**
9: for each particle $i$ do
10:     Calculate the fitness of the current position $X_{ij}^t$ using the objective function
11: end for
12: **Update personal best (Pbest) and global best (Gbest):**
13: for each particle $i$ do
14:     if fitness of $X_{ij}^t$ is better than $P_{ij}^t$ then
15:         Set $P_{ij}^t = X_{ij}^t$
16:     end if
17: end for
18: Identify the global best particle $P_g^t$ based on the best fitness among all particles
19: **Update particle positions and velocities:**
20: for each particle $i$ do
21:     Update the velocity:
            $V_{ij}^{t+1} = w \cdot V_{ij}^t + c_1 R_1^t (P_{ij}^t - X_{ij}^t) + c_2 R_2^t (P_g^t - X_{ij}^t)$
22:     Update the position:
            $X_{ij}^{t+1} = X_{ij}^t + V_{ij}^{t+1}$
23:     Ensure that the new position $X_{ij}^{t+1}$ stays within the problem's bounds
24: end for
25: **Repeat until termination condition is met:**
26: Either maximum iterations or convergence criteria are met
27: **Return:** Global best position $P_g^t$ as the solution
```

factors, commonly known as the cognitive and social parameters. Following the update and evaluation of particles, best positions (memory) are also updated at each iteration. Thus, at iteration *t+1*, the new best position of x_i is defined as follows:

$$\mathbf{P}_i^{t+1} = \begin{cases} X_i^{t+1}, & \text{if } f(X_i^{t+1}) \leq f(P_i^t), \\ P_i^t & otherwise. \end{cases} \tag{2.10}$$

Once the index P_g^t is updated, the optimal solution is established, when a PSO iteration is finished. Algorithm 1 illustrates the pseudocode of PSO operates. The global best PSO (*gbest*-PSO) and local best PSO (*lbest*-PSO) techniques are the two widely used approaches in the literature for figuring out the global best solution of PSO. While the *lbest*-PSO approach only allows a particle to be directed by a fitter particle in its local area, the *gbest*-PSO method allows the location of the particles to be affected by the particle with the more effective fitness value of the whole swarm. Similar to GA, PSO has been successfully used to solve issues across many different fields.

2.2.3 Binary Particle Swarm Optimization (BPSO)

Kennedy and Eberhart developed the first binary PSO to solve the discrete/ combinatorial optimization problems [4]. The BPSO significantly extended the basic concept of the original PSO by using the sigmoid function to convert the velocity value from continuous space in binary space.

- **Particle Representation**: Every particle has a binary string representation, with each bit denoting a decision variable. The binary string can, for instance, indicate which features are chosen (1) or rejected (0) in a feature selection issue.

- **Velocity Update**: Particles update their velocity in the continuous form of PSO by adding their prior velocity, personal best position, and global best position, all of which are weighted. This idea is modified in BPSO to regulate the probability of a particle in the binary space changing from 0 to 1. At each iteration of this BPSO, the velocity v_i of the *i*-th particle is changed by Equ. 2.8.

- **Position Update**: A probabilistic function is applied to update each particle's position once the velocity has been modified. The particle goes to position 1 if the new velocity exceeds a threshold, which is typically 0.5; if not, it remains at 0. The position of the *i*-th particle is changed by Equ. 2.11.

$$X_{ij}^{t+1} = \begin{cases} 0 & \text{if } rand() \geq S_T(V_{ij}^{t+1}), \\ 1 & \text{otherwise}, \end{cases} \tag{2.11}$$

Algorithm 2 Pseudocode for BPSO.

```
1: **Initialize the swarm:**
2: Set the number of particles $N$
3: Set the number of particle Size $noV$
4: Initialize position $X_{ij}^t$ and velocity $V_{ij}^t$ for each particle randomly within a predefined range
5: Initialize personal best position $P_{ij}^t$ and global best position $P_g^t$
6: Set the maximum number of iterations $Max\_Iter$
7: Set the parameters: inertia weight $w$, cognitive coefficient $c_1$, and social coefficient $c_2$
8: **Evaluate the fitness of each particle:**
9: for each particle $i$ do
10:     Calculate the fitness of the current position $X_{ij}^t$ using the objective function
11: end for
12: **Update personal best (Pbest) and global best (Gbest):**
13: for each particle $i$ do
14:     if fitness of $X_{ij}^t$ is better than $P_{ij}^t$ then
15:         Set $P_{ij}^t = X_{ij}^t$
16:     end if
17: end for
18: Identify the global best particle $P_g^t$ based on the best fitness among all particles
19: **Update particle positions and velocities:**
20: for each particle $i$ do
21:     Update the velocity:
            $V_{ij}^{t+1} = w \cdot V_{ij}^t + c_1 R_1^t (P_{ij}^t - X_{ij}^t) + c_2 R_2^t (P_g^t - X_{ij}^t)$
22:     Update the position using a binary transfer function (e.g., threshold function):
            $X_{ij}^{t+1} = \begin{cases} 1 & \text{if } V_{ij}^t > S(\mathbf{V}_{ij}^{t+1}), \\ 0 & \text{if } otherwise \end{cases}$
23:     Ensure that the new position $X_{ij}^{t+1}$ stays within the problem's bounds
24: end for
25: **Repeat until termination condition is met:**
26: Either maximum iterations or convergence criteria are met
27: **Return:** Global best position $P_g^t$ as the solution
```

where t is the current iteration and $S_T(v_{ij}^{t+1})$ is a sigmoid transfer function that reflects the chance (in the range [0,1]) of a bit in a binary string taking the value 0 or 1:

$$S_T(v_{ij}^{t+1}) = \frac{1}{1+e^{-v_{ij}^{t+1}}}. \tag{2.12}$$

- **Global and Personal Bests**: Similar to normal PSO, each particle records its own best location (personal best,p_{ij}^{t}) and the best place the swarm as a whole finds (global best, P_{gj}^{t}). The ideal answer discovered by the whole population is best represented by the global best.

  Algorithm 2 shows the functioning of BPSO in pseudocode. Binary particle swarm optimization (BPSO) is a well-known metaheuristic technique which is widely used for the various COPs [4]. However, it has been found that BPSO appears to lack the exploration capabilities required to produce high-quality solutions [7,8]. To address this issue, other BPSO variations have been created. However, existing BPSO variations [7,8] continue to struggle with maintaining a reasonable balance between exploration and exploitation capability. Therefore, they frequently failed to provide a high-quality solution at the final stage of the run.

2.3 Dataset

A dataset is a well-known structured collection of data that may be utilized for a variety of analytical and computational tasks. Typically, the data is arranged in rows and columns. It provides the framework for decision-making, machine learning, and data analysis. In a dataset, each column denotes a particular property or feature, and each row represents a single observation or occurrence. A dataset's objective is to offer a reliable and large availability of data for finding developments, implementing theories to the test, developing models, and making decisions. The KDD 99 data sets are briefly described in this section. You may get more specific details regarding these data sets in [9].

2.3.1 KDD CUP 99 Dataset

The most innovative and valuable resource for intrusion detection research is the Knowledge Discovery and Data Mining (KDD) Cup 99 datasets, which were first introduced in 1998. KDD refers to the family of data sets that comprises DARPA, KDD CUP 99, and NSL_KDD. These kinds of tasks have been widely used, mostly in data-stream research, machine learning, and intrusion detection. DARPA used network traffic recordings from the 1998 dataset to build the KDD'99 dataset in 1999. For each network connection, 41 features are preprocessed. As indicated in 2.1, the features in the KDD'99 dataset are divided into four groups: Basic Features (1 to 9), Content Features (10 to 22), Time based traffic features (23 to 31), and Host based traffic features (32 to 41). The 4,94,021 records in KDD'99 make it greater than other databases.

Table 2.1: Description of 41 Features in KDD Cup 1999 Dataset

Attribute No.	**Feature**	**Description**
1	Duration	The total duration of the connection between the source and destination.
2	Protocol Type	The type of protocol used for the communication (e.g., TCP, UDP).
3	Service	The specific service employed by the destination network (e.g., HTTP, FTP).
4	Flag	Indicates whether the connection status is normal or has an error.
5	Src Bytes	The amount of data (in bytes) sent from the source to the destination.
6	Dst Bytes	The amount of data (in bytes) sent from the destination to the source.
7	Land	Indicates if the source and destination have the same port number and IP address (1 for yes, 0 for no).
8	Wrong Fragment	The total number of incorrectly fragmented packets in the connection.
9	Urgent	The number of urgent packets (those with the urgent flag enabled).
10	Hot	The number of "hot" conditions, meaning attempts to access sensitive system directories.
11	Num Failed Logins	The count of failed login attempts during the connection.

Attribute No.	**Feature**	**Description**
12	Logged In	Denotes the success of the login attempt (1 for successful, 0 for failed).
13	Num Compromised	The number of times security was compromised during the connection.
14	Root Shell	Indicates if a root shell was acquired (1 for yes, 0 for no).
15	Su Attempted	Whether the 'su root' command was executed (1 if attempted, 0 if not).
16	Num Root	The total number of root-level operations performed during the connection.
17	Num File Creations	The number of file creation operations made.
18	Num Shells	The number of times a shell prompt was opened during the connection.
19	Num Access Files	The number of times access control files were modified or accessed.
20	Num Outbound Cmds	The count of commands sent outbound during an FTP session.
21	Is Host Login	Whether the login was performed by a root or admin user (1 for yes, 0 for no).
22	Is Guest Login	Indicates if the login was made as a guest user (1 for yes, 0 for no).
23	Count	The number of connections made to the same destination host.
24	Srv Count	The number of connections made to the same service on the destination host.
25	Serror Rate	The percentage of connections with specific error flags (s0, s1, s2, or s3) among those in the count.
26	Srv Serror Rate	The percentage of connections with specific error flags (s0, s1, s2, or s3) among those in the srv count.

Attribute No.	**Feature**	**Description**
27	Rerror Rate	The percentage of connections that were flagged as rejected (REJ) among the connections in the count.
28	Srv Rerror Rate	The percentage of connections flagged as rejected (REJ) among those in the srv count.
29	Same Srv Rate	The proportion of connections to the same service, out of all connections in the count.
30	Diff Srv Rate	The proportion of connections to different services, out of all connections in the count.
31	Srv Diff Host Rate	The percentage of connections made to different destination machines, from those in the srv count.
32	Dst Host Count	The number of connections made to the same destination IP address.
33	Dst Host Srv Count	The number of connections made to the same service (port number) on the destination.
34	Dst Host Same Srv Rate	The percentage of connections made to the same service, among those in the dst host count.
35	Dst Host Diff Srv Rate	The percentage of connections made to different services, among those in the dst host count.
36	Dst Host Same Src Port Rate	The percentage of connections made from the same source port, among those in the dst host srv count.
37	Dst Host Srv Diff Host Rate	The percentage of connections made to different destination machines, from those in the dst host srv count.
38	Dst Host Rerror Rate	The percentage of connections with the rejected flag (REJ), among the connections in the dst host count.
39	Dst Host Serror Rate	The percentage of connections with error flags (s0, s1, s2, or s3), among the connections in the dst host count.

Attribute No.	**Feature**	**Description**
40	Dst Host Same File Rate	The proportion of connections that accessed the same files, among those in the dst host count.
41	Dst Host Same Cmd Rate	The proportion of connections that used the same commands, among those in the dst host count.

Table 3.3 and Table 3.2 lists the four primary types of attacks: DoS, R2L (unauthorized access from a remote machine), U2R (unauthorized access to root), and Probe. To find network traffic intrusions, a variety of data mining techniques have been deployed to the KDD'99 dataset.

2.3.2 Why KDD CUP 99 Dataset?

The KDD'99 dataset is one of the oldest and most well-known standards for assessing intrusion detection systems (IDS), which is why researchers utilize this dataset. It is a desirable option for creating and evaluating machine learning algorithms due to its size, wide variety of attack types, and thorough feature representation. Researchers may concentrate on increasing detection accuracy and reducing false positives by using the dataset's organized framework for evaluating the performance of various IDS algorithms under uniform circumstances. Furthermore, by including several attack types like DoS, Probe, U2R, and R2L, it provides a wide range of challenges that aid in evaluating a model's resilience to various threat situations.

The KDD'99 dataset is still widely used despite its age, since it is easy to use and accessible, which makes it perfect for preliminary cybersecurity research investigations. It gives academics a place to test their methods before using them on greater complexity and practical datasets. Its use also makes it easier to compare the past with previous research, which enables researchers to show how IDS approaches have advanced. The dataset's widely used features and well-documented characteristics guarantee its continuous usefulness for benchmarking and prototyping novel concepts in the field of network security, despite its shortcomings, which include redundancy and obsolete attack types.

Chapter 3

Results

One of the most widely used datasets for network intrusion detection research is the KDD Cup 1999 dataset. It includes information from a military network simulation, where each record denotes a link between two devices. Each connection must be categorized as either typical or a component of an assault.

3.1 Types of Intrusion in KDD Cup 99 Dataset

Each record in the KDD Cup 1999 dataset represents a network connection and is classified as either normal (no intrusion) or as one of various attack types. DoS (Denial of Service), R2L (Remote to Local), U2R (User to Root), and Probe (Information Gathering) are the four main classifications into which these assaults fall. Below are the intrusion types in the KDD Cup 99 dataset, categorized into different types of attacks: DoS (Denial of Service), R2L (Remote to Local), U2R (User to Root), Probe (Information Gathering), and Normal (No Attack).

Table 3.1: Types of Intrusion in KDD Cup 99 Dataset

Attack Type	**Category**
Back	DoS (Denial of Service)
Land	DoS (Denial of Service)
Neptune	DoS (Denial of Service)
Smurf	DoS (Denial of Service)
Teardrop	DoS (Denial of Service)

Attack Type	**Category**
Pod	DoS (Denial of Service)
Guess_passwd	R2L (Remote to Local)
Ftp_write	R2L (Remote to Local)
Imap	R2L (Remote to Local)
Phf	R2L (Remote to Local)
Sendmail	R2L (Remote to Local)
Buffer_overflow	U2R (User to Root)
Loadmodule	U2R (User to Root)
Rootkit	U2R (User to Root)
Perl	U2R (User to Root)
Tfp	U2R (User to Root)
Satan	Probe (Information Gathering)
Mscan	Probe (Information Gathering)
Nmap	Probe (Information Gathering)
Ipsweep	Probe (Information Gathering)
Sweeps	Probe (Information Gathering)
Normal	No Attack

3.2 Parameter Set Up

Our research uses a particle population of 100 and the Binary Particle Swarm Optimization (BPSO) method for feature selection. The following is how the algorithm parameters are set up: With a dynamic range between a maximum (w_{max}) of 0.4 and a minimum (w_{min}) of 0.9, the cognitive coefficient (c_1) and social coefficient (c_2) are both set to 2. The inertia weight (w) is also initialized at 2. To limit the swarm's capacity for exploration, the maximum particle velocity (V_{max}) is set at 4. The KDD Cup 99 dataset, of which 30% as Table 3.2 is set aside for training and 70% as Table 3.3 for testing, is used to carry out the feature selection procedure. In order to enhance the model's performance in network intrusion detection tasks, this configuration seeks to determine the appropriate attributes.

3.2.1 Training Set

The distribution of the various attack types in the KDD Cup 1999 test set is displayed in Table 3.2, along with the approximate number and percentage of each category. Denial of Service (DoS) attacks are the most common type of attack in the test set, accounting for 73.90% of the data, or around 109,302 occurrences. Approximately 28,905 records, or 19.48% of the test set, are normal traffic. About 7,703 items, or 5.20% of the dataset, are Remote to Local (R2L) attacks. With around 1,988 records, probe (information gathering) assaults make up 1.34% of all attacks. Just 0.07% of the test set, or about 208 records, contain User to Root (U2R) attacks, making them incredibly uncommon. The test set has a very unbalanced distribution

Table 3.2: Training Set Distribution

Attack Type	**Percentage (%)**	**Count (approx.)**
Normal	19.48%	28,905
DoS	73.90%	109,302
Probe	1.34%	1,988
R2L	5.20%	7,703
U2R	0.07%	208
Total	100%	148,206

with a total of about 148,206 records. DoS attacks predominate in the test set, much like in the training set. In contrast, other attack types—particularly U2R—are far less common, making it difficult to identify these uncommon attack types during model training and evaluation.

3.2.2 Test Set

The KDD Cup 1999 training set's attack types are distributed according to the Table 3.3, which also displays the approximate number and proportion of each category. The dataset is dominated by normal traffic, which makes up 75.61% of all records, or around 261,167 occurrences. Denial of Service (DoS) attacks are the second most common category, accounting for around 79,522 entries, or 23.00% of the data. At 1.29%, or around 4,457 records, probe (information gathering) attacks make up a lower percentage of the dataset. Even less common are remote to local (R2L) attacks, which account for approximately 0.093% of the sample and only occur in about 321 cases. User to Root (U2R) attacks are the least frequent, accounting for about 0.005% of all records, or around 17 entries.

The dataset has 345,815 items in total, and the distribution of attack types is obviously unbalanced. The other attacks, especially R2L and U2R, are substantially less common, even though normal traffic makes up the great majority. This is crucial for understanding the difficulties in training models to successfully detect these uncommon attack types.

Table 3.3: Test Set Distribution

Attack Type	**Percentage (%)**	**Count (approx.)**
Normal	75.61%	261,167
DoS	23.00%	79,522
Probe	1.29%	4,457
R2L	0.093%	321
U2R	0.005%	17
Total	100%	345,815

Chapter 4

Result & Discussion

The results of our proposed research methodology show how well a classification model performed, where:

- True Negatives (TN) = 67,880: These are the cases in which the model classified normal traffic as normal, i.e., properly predicted the negative class. For typical traffic, the model produced 67,880 accurate predictions.
- False Positives (FP) = 18: These are the cases in which the model misclassified regular traffic as an attack, i.e., mispredicted the positive class. 18 of the model's predictions were wrong, misclassifying regular traffic as an attack.
- False Negatives (FN) = 46: These are the cases in which the model misclassified objects as normal because it was unable to identify a genuine attack. 46 attack occurrences were overlooked by the model, which mislabeled them as typical traffic.
- True Positives (TP) = 277,871: These are the cases in which the model detected attacks as attacks and predicted the positive class. For attack traffic, the model produced 277,871 accurate predictions.

According to the result, Our proposed model is doing well in accurately categorizing regular traffic and identifying attacks, especially considering the low values for FP and FN. Overall, the model is producing accurate predictions, as evidenced by the high True Positives and True Negatives. Further improvements or testing on boundary instances, where specific attack types or everyday traffic could be misclassified, could still improve the model.

4.1 Performance Evaluation

Given the confusion matrix values:

True Negatives (TN) = 67,880, False Positives (FP) = 18,False Negatives (FN) = 46, True Positives (TP) = 277,871

We can calculate the following metrics using these parameter:

Accuracy

Accuracy is the proportion of correctly classified instances (both normal and attack) out of the total instances:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{277,871 + 67,880}{277,871 + 67,880 + 18 + 46} = \frac{345,751}{345,815} \approx 99.98\%$$

Precision

Precision is the proportion of correctly identified attack instances out of all instances that were predicted as attacks:

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{277,871}{277,871 + 18} \approx 99.99\%$$

Recall

Recall is the proportion of correctly identified attack instances out of all actual attack instances:

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{277,871}{277,871 + 46} \approx 99.98\%$$

Specificity

Specificity is the proportion of correctly identified normal instances out of all actual normal instances:

$$\text{Specificity} = \frac{TN}{TN + FP} = \frac{67,880}{67,880 + 18} \approx 99.97\%$$

F1-Score

The F1-score is the harmonic mean of Precision and Recall, providing a single metric that balances both concerns:

$$\text{F1-Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \approx 99.99\%$$

4.2 Comparison of Performance

According to 4.1, With an accuracy of 98.6%, the PSO+DT approach—which blends Particle Swarm Optimization with a Decision Tree—shows strong overall performance. But at 75.3%, its accuracy is significantly lower, suggesting that it is more likely to produce false positives, which might be an issue in some applications. Although it has a high specificity (98.8%), which indicates that it successfully detects negative situations, its F1-Score of 81.8% indicates that accuracy and recall are not as well matched. PSO+DT is a respectable technique overall, but its lack of accuracy and F1-Score limits its use in high-stakes situations where false positives might be expensive.

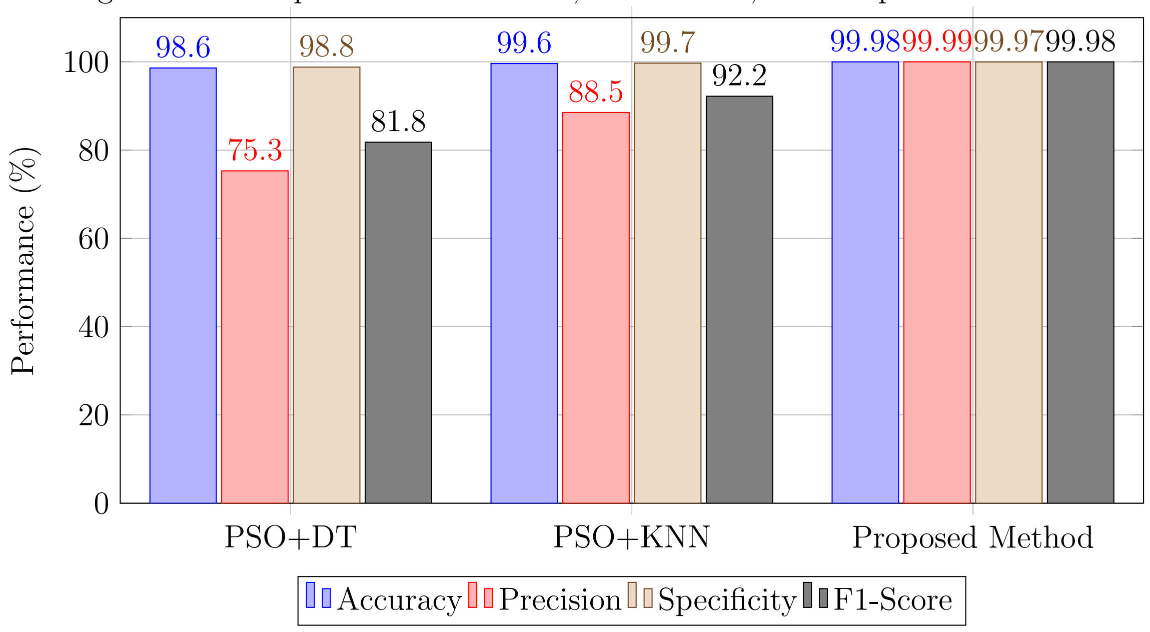
Table 4.1: Comparison of PSO+DT, PSO+KNN and Proposed Method

Method	**Accuracy (%)**	**Precision (%)**	**Specificity (%)**	**F1-Score (%)**
PSO+DT	98.6	75.3	98.8	81.8
PSO+KNN	99.6	88.5	99.7	92.2
Proposed Method	99.98	99.99	99.97	99.98

However, with an accuracy of 99.6% and a significantly greater precision of 88.5%, the PSO+KNN approach outperforms PSO+DT. According to this, PSO+KNN is far more successful at reducing false positives, which is important for applications that need a high level of dependability. Moreover, its 99.7% specificity shows that it is also quite dependable in detecting negative instances. PSO+KNN is a more efficient choice than PSO+DT because it achieves a far better balance between accuracy and recall, as seen by its F1-Score of 92.2%. With nearly flawless scores on all metrics—accuracy, precision, specificity, and F1-Score—the Proposed Method, however, performs significantly better than both PSO+DT and PSO+KNN.

The histogram as in Figure 4.1 that analyzes the performance of PSO+DT, PSO+KNN, and the proposed technique makes it evident how well each approach performs in terms of the following important metrics: F1-Score, Accuracy, Precision,

Figure 4.1: Comparison of PSO+DT, PSO+KNN, and Proposed Method



and Specificity. With an accuracy of 98.6% and specificity of 98.8%, PSO+DT performs admirably, proving that it is capable of accurately detecting negative instances. Its overall dependability is harmed by its relatively poor accuracy of 75.3%, which indicates that it has trouble reducing false positives. The method's shortcomings in attaining a balanced categorization are reflected in the F1-Score of 81.8%, which further emphasizes the imbalance between accuracy and recall.

On the other hand, PSO+KNN achieves 99.6% accuracy, 88.5% precision, and 99.7% specificity, outperforming PSO+DT in every way. This approach's increased accuracy and F1-Score of 92.2% show that it can more accurately categorize both positive and negative situations while lowering false positives. However, with nearly flawless results on all metrics- 99.98% accuracy, 99.99% precision, 99.97% specificity, and 99.98% F1-Score,-the Proposed Method much outperforms both PSO+DT and PSO+KNN. This outstanding result demonstrates that the Proposed Method is the most dependable and well-rounded strategy, providing better classification accuracy and a significantly greater degree of consistency in accurately recognizing both positive and negative situations with the least amount of error.

Chapter 5

Conclusion

When applied to the KDD CUP 99 dataset, the suggested particle grouping method for feature selection combined with Binary Particle Swarm Optimization (BPSO) showed remarkable results. By using the cooperative dynamics of grouped particles, this novel approach successfully tackles the problem of choosing the best subset of characteristics. The method greatly improves the convergence towards discriminative and meaningful feature subsets by directing the search process through grouped interactions, which lowers redundancy and boosts computing efficiency.

The outcomes demonstrate the effectiveness of this method in managing the high-dimensional data complexity of the KDD CUP 99 dataset. The impressive accuracy and smaller feature set highlight how well particle grouping works to retain important characteristics while eliminating unnecessary ones. Better classification performance results from this, proving its usefulness in intrusion detection systems and other fields that need reliable and effective feature selection techniques.

In summary, the particle grouping-based feature selection method integrated into BPSO not only performed better than conventional techniques, but also established a standard for feature selection research going forward. This approach is a potential tool for tackling feature selection issues in a variety of datasets due to its flexibility and scalability. The topic of feature selection and optimization might be further advanced by future research that investigates its application to other datasets and extends the approach to address dynamic and real-time data contexts.

Chapter 6

References

Bibliography

[1] Alzubi, Q., Anbar, M., Sanjalawe, Y., Al-Betar, M. & Abdullah, R. Intrusion detection system based on hybridizing a modified binary grey wolf optimization and particle swarm optimization. *Expert Systems With Applications.* **204** pp.117597 (2022)

[2] Holland, J. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. (MIT press,1992)

[3] Kennedy, J. & Eberhart, R. Particle swarm optimization. *Proceedings Of ICNN'95-international Conference On Neural Networks.* **4** pp. 1942-1948 (1995)

[4] Kennedy, J. & Eberhart, R. A discrete binary version of the particle swarm algorithm. *1997 IEEE International Conference On Systems, Man, And Cybernetics. Computational Cybernetics And Simulation.* **5** pp. 4104-4108 (1997)

[5] Dorigo, M., Maniezzo, V. & Colorni, A. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions On Systems, Man, And Cybernetics, Part B (Cybernetics).* **26**, 29-41 (1996)

[6] Karaboga, D. & Others An idea based on honey bee swarm for numerical optimization. (Technical report-tr06, Erciyes university, engineering faculty, computer . . . ,2005)

[7] Mirjalili, S. & Lewis, A. S-shaped versus V-shaped transfer functions for binary particle swarm optimization. *Swarm And Evolutionary Computation.* **9** pp. 1-14 (2013)

[8] Bansal, J. & Deep, K. A modified binary particle swarm optimization for knapsack problems. *Applied Mathematics And Computation.* **218**, 11042-11061 (2012)

[9] Lippmann, R., Haines, J., Fried, D., Korba, J. & Das, K. The 1999 DARPA off-line intrusion detection evaluation. *Computer Networks*. **34**, 579-595 (2000)