Enhanced Particle Swarm Optimization-Based Approach to Firewall Optimal Rule Reordering

Dhwani Hakani and Palvinder Singh Mann*

Gujarat Technological University, Ahmedabad, Gujarat, India
Email: adf_dhwani@gtu.edu.in (D. H.); asso_psmaan@gtu.edu.in (P. S. M.)

*Corresponding author

Manuscript received August 7, 2024; revised September 9, 2024; accepted November 13, 2024; published November 20, 2024

***Abstract*—Firewalls are required to ensure that only trustworthy packets are sent back and forth across the network in order to provide secure network communication. Firewalls employ the rules that network administrators establish to control which packets are allowed access to an organization's private network in order to enforce security regulations. By classifying packets, network devices can determine how incoming packets behave. A greater communication delay is caused by a higher rule count since it is achieved by a linear search on a list of categorisation rules. The goal of Optimal Rule Ordering (ORO), a generalisation of the issue where the latency is minimised while keeping the classification strategy, is to find the optimal rule sequence. This research suggests a dual approach for reordering the firewall rules using optimization. This research suggests a dual approach for reordering the firewall rules using optimization. In the first approach, the firewall rules are arranged according to the precedence relation using a probability-based algorithm. The firewall rules are then rearranged using the optimization-based technology known as Particle Swarm Optimization (PSO). Firewalls may be optimized to function better and filter packets more effectively by fine-tuning their rules.The performance analysis of the proposed method is extended by looking at the results obtained using a precise optimization strategy. This study presents a method for rearranging more complex scenarios that work better. The proposed method consists of two algorithms: the first finds the ideal firewall rule order using a probability-based approach, and the second finds the optimal solution using a PSO: i) An ideal firewall rule order via a probability-based approach, and ii) An optimal solution using a PSO-based approach.**

***Keywords*—firewall, Optimal Rule Reordering (ORO), optimization, rule reordering, probability, Particle Swarm Optimization(PSO)**

I. Introduction

Firewalls are computer systems that mediate access between two or more network segments or computer hardware and software systems or between a network interface and the rest of the computer system hosting it [1]. The two main purposes of firewalls are to allow authorized network traffic to pass through and to block unauthorized network traffic [2]. The process of controlling network packets according to a logical set of rules—a filtering policy—is known as packet filtering. Due to the constantly changing network architecture and the constant requests for new services from users, systems administrators must regularly add, update, and delete filtering rules from policies [3]. As a result, the filtering policy gets more complicated after every change the user makes, and the rule order may not be ideal. It has spurred the scientific community to concentrate on other strategies for dependable optimization techniques. The most often used type of firewalls are rule-based firewalls [4], and since typically triggered rules are checked too frequently, an incorrect rule ordering might cause a performance bottleneck. Therefore, it is necessary to examine the traffic characteristics to determine which regulations are outdated or have not been used for an extended period [5]. To acquire the smallest number of packet matches and prevent significant performance reduction from traffic anomalies, firewall rules must be arranged adaptively. It has been demonstrated that the optimal rule ordering problem (ORO) with rule dependence restrictions is NP-complete. As a result, updating the rule ordering dynamically concerning traffic volumes is expensive [6]. In the event of complex firewall security settings, classical optimization techniques may not discover a solution for a long time. As a result, heuristic algorithms—also known as heuristics—have been used. Heuristics [7], on the other hand, typically seek to arrive at a suitable sub-optimal solution more quickly. Generally, a heuristic's departure from a solution that would result in significant improvements increases with decreasing execution time. To better grasp their true effectiveness, it is crucial to compare the outcomes of heuristic recommendations with precise optimization techniques [8]. Particle Swarm Optimization (PSO) is a widely used heuristic approach for solving NP-hard combinatorial optimization problems [9]. PSO has been effectively used to solve a variety of related combinatorial optimization issues, such as the traveling salesman problem [10], scheduling [11], and engineering challenges [12]. This drives our research into examining using a PSO-based heuristic technique to solve the firewall optimization problem. The ORO problem is addressed by the approach presented in this study, which also uses this heuristic. However, the PSO will not be utilized alone because, as recent research has shown, it is more successful when combined with other algorithms to provide even greater optimal performance. Specifically, the algorithm described in this study uses a PSO as the second step in a

dual approach. Finding the order of the rule sequence based on probability is the first step in the process. Once the first stage results are received, the PSO maintains the order of the rules and organizes the rule list within the previously optimally sorted sequence. Consequently, the PSO searches for the best location for the rule sequence. To tangibly verify the suggested algorithm's efficacy, it presents a comparison using advanced heuristics and a precise optimization technique. Importance of Reordering Firewall Rules:

- Enhanced Security Specificity: Placing specific rules before general ones ensures that critical security measures are applied first, reducing the risk of unauthorized access.
  Minimized Attack Surface: Effective ordering can help block malicious traffic before it reaches sensitive areas of the network. Improved Performance
- Efficiency: By prioritizing frequently used rules or those that filter out high volumes of traffic, the firewall can process packets more quickly, reducing latency.
  Reduced Load: Streamlined rule processing lowers the overall load on the firewall, allowing it to handle more connections simultaneously without degradation.
  Simplified Management
- Clarity: A well-organized rule set is easier to read and maintain, enabling quicker identification of issues and necessary updates.
  Easier Auditing: A clear structure helps in reviewing rules during audits, making it simpler to verify compliance with security policies.

Hence, reordering firewall rules is critical for both security and performance. Properly structured rules enhance protection against threats, streamline traffic handling, and simplify ongoing management, ultimately leading to a more robust and efficient network environment.

The remaining portion of this paper is structured as follows: Section II explains the related work of various firewall rule optimization with rule reordering. Section III explains authors contribution. The proposed firewall reordering is explained in Section IV. Section V explains results and discussion. The performance of the proposed approach is described in Section V, and Section VI concludes the research paper.

II. Related Work

This section explains the various approaches related to firewall rule optimization with reordering and PSO algorithm for addressing NP-Hard problems. NP-hardness is a computational complexity concept that measures the difficulty of solving problems in polynomial time. All NP-complete problems are also NP-hard.

Hamed *et al.* [12] describe a novel method for optimizing firewall filtering settings using Internet traffic features. The method dynamically optimizes the ordering of packet filtering rules in real time in response to traffic conditions by using actively computed statistics. The optimization algorithm considers both the significance of the rule in traffic matching and its dependence on other rules. Using Directed Acyclical Graphs (DAGs) to express firewall policy, Tapdiya *et al.* [13] present a novel rule-sorting technique. This algorithm first uses the list of constraints between rules to compute the DAG. Given such a DAG, the list of reachability sets connected to every rule is derived. DAG base optimization is explained in [14]. The average activation frequency of the rules in each set is then determined, giving each set a weight. In the end, a weighted system that considers rules close to the rule that identifies the generic set is used to sort each set in ascending order. In [15], Fulp suggested a straightforward bubble sort-like heuristic algorithm based on admissible swaps of neighbouring rules and described the precedence relations among rules as a DAG. Neji *et al.* [16] present a novel optimization method that makes optimization simple, safe, and optimal by introducing new matrix-based evaluation metrics. A collection of unique matrices, including the Dependency, Reordering, and Grouping matrices, are also used to provide an evaluation performance. Every matrix has a corresponding factor in [0,1], and the defined factors are presented to quantify the reordering possibilities and measure the filtering methods' complexity. Using an innovative concept known as the Swapping Window, Mohan *et al.* [17] provide an effective and straightforward approach for improving the firewall's rule order. The author also presents a novel adaptive approach for estimating the statistics of multinomial observations that appear in a batch model. The approach adapts the Stochastic Learning Weak Estimation (SLWE) and can handle nonstationary settings. Mohan *et al.* [18] expanded Fulp's technique by identifying permissible switching windows—rule sequences in which the first and final rules can be switched without going against precedence restrictions. The matching frequencies are assumed to be independent of the rule position in the sequence in all of these works. Using statistical analysis, Kadam *et al.* [19] present an adaptive cross-domain firewall policy optimization technique that maintains policy confidentiality. The author suggests a method that uses network statistics to determine the rules' order dynamically. The method finds the order of the rules in the rule set to enhance system performance and identify and eliminate redundant rules. Two tasks are involved in the optimization process: First, while maintaining the privacy of each firewall, work together to lower the overall number of rules. Secondly, determine the rule set's order by utilizing network usage information. According to Harada *et al.* [20], an oriented tree can be formed by the precedence relation graph of an input rule list if it is limited to an inclusive rule list. By reconstructing the input rule list as an inclusive rule list, the author's rule reconstruction approach heuristically optimizes it. The Relaxed Optimal Rule Ordering (RORO) problem is examined by Harada *et al.* [21], where rules can be substituted provided their actions are identical. The weight of rules in RORO might fluctuate as they change around. A zero-suppressed binary decision diagram is the method the author suggests employing to determine the weights. This approach precisely calculates the latency and

computes a rule list that guarantees less latency than in numerous other traditional algorithms. Fixed Sub-Graph Merging (SGM) is an alternate form of SGM that Fuchino *et al.* [22] propose. Concerning the allocatable elements, this algorithm arranges the items within the chosen subgraph, selecting the heaviest (even if single). A simulated annealing technique for RORO based on policy violation, as opposed to overlap or dependency relations on rules, is proposed by Harada *et al.* [23]. Using a Zero-suppressed binary Decision Diagram (ZDD), the method concentrates on an ordering that fulfils the policy but does not satisfy the dependence relation. A unique method for resolving the issue of the best arrangement of filtering policies in a firewall is developed by Coscia *et al.* [24]. The method combines two heuristics for managing mutually dependent policies and applies a genetic algorithm. It offers improved efficiency from the perspective of time processing and filtering action speed. Coscia *et al.* [25] present a novel two-stage approach that optimally reorders firewall security rules to minimize packet classification latency. To identify the best ordering for the limited rules, the first step uses a novel topological sorting method that considers the inter-packet arrival time—which generally follows Zipf's law—influences rule activation frequencies. A genetic algorithm is used in the second stage to determine the best order for each rule in the list. An investigation on the computational cost of permissible rule ordering and its greedy algorithm was carried out by Fuchino *et al.* [26]. By examining computational complexity, their research highlighted the NP-hardness of some rule-ordering procedures and illuminated the difficulties associated with rule optimization. An enhanced method for assisted firewall anomaly identification and resolution is put forth by Bringhenti *et al.* [27]. This method only solves suboptimizations automatically; it communicates with users by asking direct questions about conflict resolution because automatic conflict resolution can result in undesirable configurations. To lessen the administrative burden, the method also minimizes the number of interactions necessary and uses satisfiability checking approaches to produce a correct-by-construction outcome. Pizzato *et al.* [28] provides an effective technique to minimize the processing duration for reconfiguration while offering an automated, formally proper, and efficient placement and configuration of the essential network security functions. This method incurs a significant computing load, with the duration varying from a few seconds to several minutes, contingent upon the intricacy of the network. Chao *et al.*, [29] propose a two-phased technique to reduce the number of filtering rules in firewalls while maintaining the original filtering effects for IPv6 traffic in high-speed IoT networks. The mechanism incorporates a novel split-and-merge technique to efficiently divide and combine the traffic filtering regions of firewall rules, thereby reducing the total number of firewall rules.

The ORO problem can be solved with a suboptimal solution using any of the strategies above. While some are quite fast, their classification latency minimization performance is unclear because no exact optimization approach that can yield a global optimal solution is compared with them. Regardless of the complexity of the problem, quick reordering is required in real-life scenarios; however, the ultimate objective is to maximize the FW packet processing time. Comparing the performance attained by precise optimization is therefore crucial. A new algorithm to solve the ORO problem is presented in this study. Table 1 shows the summary of firewall rule ordering.

Table 1. Summary of firewall rule ordering

Ref	Description	Technique Used	Remarks
[12]	Dynamic firewall rule reordering	Statistical packet matching	The method poorly manages the intermittent nature of packet flows.
[13]	DAG-based rule ordering (SubGraph Mergine)	Heuristic sorting - Sub-Graph Merging algorithm	It efficiently finds the optimal rule order.
[14]	DAG-based firewall rule optimization	Rules sorting (based on precedence relationships)	Finding a solution is a time-consuming task.
[15]	Optimal filtering rule ordering	Matrix-based rule reordering	It has great potential to simplify, optimize, and secure optimization.
[16]	Dynamic ordering firewall rules.	Swapping Window-based rule ordering	Unnecessary overhead.
[17]	Network traffic statistic-based rule ordering	Rule reordering and traffic aware algorithm	Excessive delays in packet filtration might result in packet losses and session initiation denials.
[18]	Adaptive cross-domain firewall policy optimization	Statistical based technique	The sequence of rules is determined by using statistics on network usage.
[20]	Relaxed optimal rule ordering	Zero-suppressed binary decision diagram with a Heuristic algorithm	It computes the latency precisely.
[21]	Packet classification using Adjacency List	Adjacency List based approach and Fixed SubGraph Mergine	It decreases reordering time compared with the original SGM [R2]
[22]	Relaxed Optimal Rule Ordering	Simulated annealing	ZDDs are used in the procedure to determine if an order complies with the policy.
[23]	Firewall policy optimization	Genetic algorithm	It reduces the time complexity
[24]	Firewall rule ordering using two stage algorithm	Topological sorting and genetic algorithm	It reduces the error induced by uncertain operators, improving the examination potential of the entire strategy.
[25]	Relaxed optimal rule ordering	Greedy Method	It reduces the latency
[26]	Intra-firewall policy anomaly detection and resolution	Optimal approach for anomaly analysis	It provide efficient anomaly resolution and reduces the workload
[36]	Optimized firewall reconfiguration	Automatic and optimal approach	It increases computational complexity and processing time.
[37]	Firewall rule optimization	Two phase mechanism, split and merge method	It enhances the efficiency and efficacy of the mechanism.

Heuristic algorithms work well for addressing NP-Hard issues. Mor *et al.* [30] demonstrate how heuristic-achieved

solutions for this class of optimization problems are nearly identical to global solutions discovered by exhaustive search techniques. Heuristics are faster than exhaustive searches; hence, these two kinds of algorithms are not comparable in convergence speed. PSO is a popular heuristic for handling NP-Hard issues. An NP-Hard issue called the Travels Salesman issue (TSP) involves a salesman who has to visit every city and return to the starting location in the least amount of time. To discover the optimal solution for TSP, one of the most well-known NP-Hard problems in computational optimization, Abdulrahman [31] uses the most effective heuristic-based Swarm Intelligence techniques. A novel transfer learning-based PSO method (TL-PSO) for TSPs is proposed by Zheng *et al.* [32]. A city topology matching approach based on geometric similarity is first developed to match each new city subset to a historical city subset. All cities in the new and historical TSP problems are clustered into several city subsets. Next, to produce a quality initial swarm, useful optimal pathways are extracted from those matching historical city subsets and then transferred into the target region.Swarm intelligence is the collective behavior of decentralized, self-organized systems, natural or artificial. The concept is employed in work on artificial intelligence. The expression was by this token introduced. Wei *et al.* [33] suggest improved hybrid particle swarm optimization to address the Travel Salesman Problem (TSP). A probability initialization is first utilized to incorporate previous knowledge into the initialization to save significant computational power as the algorithm evolves. Furthermore, two crossover types are suggested to utilize Gbest and Pbest better to increase the algorithm's convergence accuracy and population diversity. A directional mutation is finally used to get around the typical mutation operator's randomness. Based on particle swarm optimization and genetic algorithms, Kou *et al.*'s [34] optimal patrol path design for offshore wind farms addresses the travelling salesman problem. First, the travelling salesman issue of shortest route optimization characterizes offshore wind farms' patrol routing planning problem. Second, the patrol path distance is used as the objective function, and the GA and PSO algorithms are independently simulated and validated. It is determined that the task scheduling problem is a nondeterministic polynomial time (NP)-hard problem. Randomization of the initialization of solution searching is a crucial component of these optimization techniques. Nonetheless, the performance of metaheuristic algorithms can be greatly enhanced by providing them with efficient initialized solutions. Using heuristic techniques, Alsaidy *et al.* [35] suggest an enhanced initialization of particle swarm optimization. To prevent premature convergence and speed up the convergence of standard PSO, Agarwal *et al.*'s [36] particle swarm optimization task scheduling mechanism uses an opposition-based learning technique. It is then compared to other well-known PSO-based task scheduling strategies. An approach called Improved Particle Swarm Optimization is proposed by Pirozmand *et al.* [37]. A multi-adaptive learning technique is used to reduce the execution time of the original Particle Swarm Optimization algorithm for task scheduling in the cloud computing environment. The Multi Adaptive Learning for Particle Swarm Optimization defines two types of particles in its initial population phase: ordinary particles and locally best particles. The population becomes less diverse during this phase and the chance of reaching the local optimum increases.

III. Authors' Contribution

There are two authors in this paper. This study proposes two methods for optimizing the firewall rule reordering. In the first method, a probability-based algorithm arranges the firewall rules according to the precedence relation. The firewall rules are then rearranged using the optimization-based technology known as particle swarm optimization (PSO). Firewalls can be optimized to perform better and filter packets more effectively by fine-tuning their policies. The following are the contributions of this work:

- To present a novel dual approach based on Probability and optimization to optimize the Firewall rule order.
- A novel probability distribution-based sorting technique is recommended to reduce the classification delay related to the sorted objects.
- The PSO algorithm suggests the best rule ordering, which switches rules without breaking against policy.

IV. Proposed Methodology

This section explains the proposed dual approach-based firewall rule reordering. In the first method, a probability-based algorithm arranges the firewall rules according to the precedence relation. The firewall rules are then rearranged using the optimization-based technology known as Particle Swarm Optimization (PSO).

- Probability-Based Approach: A probability-based approach in optimization typically involves using probabilistic models to make decisions or predictions about the behavior of a system. This can include: Random Sampling: Instead of exhaustively searching through all possible solutions, this approach samples solutions based on their probabilities of being optimal. It helps to explore a wide solution space while focusing on promising areas. Stochastic Methods: These techniques incorporate randomness into the search process, allowing for exploration of the solution space without getting stuck in local optima. This is particularly useful in complex landscapes where deterministic methods may fail. Adaptive Mechanisms: The probabilities can adapt based on feedback from the optimization process, allowing the algorithm to dynamically focus on more promising solutions as it learns about the landscape.
- Particle Swarm Optimization (PSO): Particle Swarm Optimization is a population-based optimization technique inspired by the social behavior of birds or fish. Here's how it works:
- Initialization A swarm of particles (potential solutions) is initialized randomly in the solution space, each with a position and velocity.

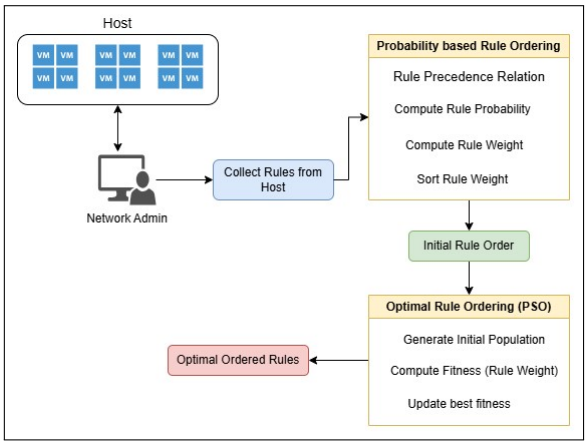

Fig. 1. Proposed architecture.

- Fitness Evaluation: Each particle's position is evaluated using a fitness function, which measures how good that solution is in terms of the optimization goal.
- Individual and Collective Learning: Each particle keeps track of its best position found so far (personal best) and shares this information with the rest of the swarm. The swarm also tracks the best position found by any particle (global best).
- Velocity and Position Update: Each particle adjusts its velocity based on its own experience and that of its neighbors, guided by the equations: New Velocity = Current Velocity + Cognitive Component + Social Component New Position = Current Position + New Velocity The cognitive component drives the particle toward its own best position, while the social component pushes it toward the global best.
- Iteration: The process repeats for a set number of iterations or until a convergence criterion is met. As the algorithm progresses, particles tend to cluster around the best solutions, exploring the search space more efficiently.

Benefits of PSO:

- Simplicity: PSO is relatively simple to implement and requires few parameters to adjust.
- Efficiency: It converges quickly to good solutions, making it suitable for many optimization problems.
- Flexibility: PSO can be applied to various types of optimization problems, including continuous, discrete, and combinatorial ones.

Hence, the probability-based approach, along with techniques like Particle Swarm Optimization, provides effective frameworks for solving complex optimization problems by balancing exploration and exploitation of the solution space. PSO, in particular, leverages collective intelligence to enhance convergence speed and solution quality. Fig. 1 shows the general architecture of firewall rule ordering.

A. Firewall Policy

Packets from one network to another are filtered by a network firewall using rules. Formally stated, a firewall policy, R, is an ordered list of rules R = (r_1, r_2,..., r_N), where N is the total number of rules in the list. An M-ary tuple of attributes or fields (fdi_1, fdi_2,...,fdi_M) and a corresponding action (ai), which can be either allow or deny, comprise of each rule r_i (for i = 1, 2,..., N). The most often utilized attributes are the protocol type, source IP address, source port number, destination IP address, and destination port number. These fields are fundamental and it provides crucial information for screening and regulating network traffic. The other firewall rule fields such as Application or Service, Protocol Flags, Packet Size, and MAC Address are infrequently utilized. Every rule attribute denotes a particular value or collection of values. Table 2 shows the sample firewall rules where Prot is Protocol, SIP is source IP, SPt is source port, DIP is Destination IP, DPt is Destination port. From Table 2, the total number of rules (N) are 10, the

Table 2. Sample firewall rules

RID	Prot	SIP	SPt	DIP	DPt	Action
1	TCP	192.4.4.44	45	192.4.4.18	53	Allow
2	TCP	192.4.4.44	45	192.4.4.18	53	Deny
3	TCP	192.4.4.28	515	192.4.4.50	45	Allow
4	TCP	192.4.4.20/40	8080	192.4.4.10/52	143	Allow
5	TCP	192.4.4.20/30	8080	192.4.4.10/50	143	Deny
6	TCP	192.4.4.38	8080	192.4.4.10/52	143	Allow
7	TCP	192.4.4.38	8080	192.4.4.10/52	143	Deny
8	TCP	192.4.4.15	80	192.4.4.11/52	45	Deny
9	TCP	192.4.4.23/35	8080	192.4.4.38	143	Deny
10	TCP	192.4.4.15	80	192.4.4.40	45	Allow

number of fields (M) are 6 (RID is not considered) For rule1, fd_{11} = TCP, fd_{12} = 192.4.4.44, fd_{13} = 45, fd_{14} = 192.4.4.18, fd_{15} = 53, fd_{16} = Allow. A packet is compared one by one, starting with the first rule in the security policy, against every rule until it finds a match—usually the first match—or reaches the end of the list when it reaches the firewall. The matching rule action is executed if one is discovered; if not, the default rule action, which is often denied, is carried out. When every attribute in a packet falls inside the range of values of the associated rule attribute, the packet is said to match the rule. It is crucial to consider the order of the rules in the list when creating a security policy since they are not always mutually exclusive—a general rule can be a superset of one or more other rules. If every matching field's intersection is not empty, then two rules are considered intersecting. Inadequate firewall performance or a breach in the security policy's integrity can arise from improper ordering. The ORO problem is restricted to optimization since the rules have a dependency relationship. It is impossible to reposition these restrictions arbitrarily without jeopardizing the security guidelines that the original FT set in place. When two rules, r_i and r_j, are given and belong to the same set R, meaning that i is not equal to j they are considered dependent if r_i and r_j match the same network traffic, r_i and r_j carry out distinct filtering action. Table 3 shows the precedence relation of Table 2.

Consider the relation [1→ 2] in Table 3. The rule r_1 must precede r_2 since both rules have the same protocol, IP address, ports and different actions.

Table 3. Precedence relation from Table 2

Rule ID	Precedence Relation
1	[2]
2	[null]
3	[null]
4	[5, 7, 9
5	[null]
6	[7]
7	[null]
8	[10]
9	[null]
10	[null]

B. Probability-Based Rule Ordering

Assume that a firewall policy consists of N rules R = (r_1, r_2,..., r_N). The rule matching Probability of each rule r_i is known to be P(r_i)= p_i. Equation (1) can be utilized to obtain this probability distribution. Since a firewall's performance depends on the rule order, choosing an order that minimizes computing cost is necessary to make a judgment. Nonetheless, regulations often intersect, and a misalignment of specific requirements may result in a security breach. Therefore, it is necessary to maintain the relative order of rules r_i and r_j if they intersect. An integer program with constraints can create an optimization problem for firewall policies. Suppose the filtering table comprises rows with numbers ranging from 1 to N, with index one assigned to the first row. It is also assumed that the original policy list's rules are indexed from 1 to N. The probability for each rule can be calculated as,

$$P\left(r_i\right) = p_i = \frac{\sum_{j=i+1}^{N} \eta_{ij}}{N} \tag{1}$$

where, η_{ij} indicate the summation of the matching field of rule r_i and r_j. It can computed as

$$\eta_{ij} = \frac{\sum_{f=1}^{M} \delta_{ij}^{f}}{M} \tag{2}$$

where, δ_{ij}^{f} is calculated as

$$\delta_{ij}^{f} = \begin{cases} 1, & \text{if } r_i{=}r_j \\ 0, & \text{Otherwise} \end{cases} \tag{3}$$

If the value of δ_{ij}^{f} is assigned 1 when the two rules r_i and r_j fields are identical; otherwise, 0. Table 4 shows the rules matching Probability.

Table 4. Rules matching probability

Rule ID	Matching Probability
1	0.4
2	0.325
3	0.486
4	0.8
5	0.52
6	0.653
7	0.53
8	0.7
9	0.2
10	0.0

Let τ_{ij} represent binary decision variables that represent whether rule i is or is not assigned to row j. It is defined as,

$$\tau_{ij} = \begin{cases} 1, \text{ if } r_i \text{ is located at } j \text{ th row in list} \\ 0, \text{ Otherwise} \end{cases} \tag{4}$$

where i and j ϵ 1,2,3,..., N . Furthermore, let ρ_{ij} specify the following precedence connection between rules r_i and r_j:

$$\rho_{ij} = \begin{cases} 1, \text{ if } r_i \text{ is precede } r_j \\ 0, \text{ Otherwise} \end{cases} \tag{5}$$

The Probability for the precedence matrix is computed as,

$$\mathrm{Pm}(i) = \frac{\sum_{j=1}^{N} \rho_{ij}}{N} \tag{6}$$

Here Pm(i) is the matching probability. This research will use the following modeling to assess the outcomes of an exact optimization strategy to solve the ORO problem.

$$\text{minimize} \sum_{i=1}^{N} \sum_{j=1}^{N} j p_i \tau_{ij} \tag{7}$$

Subject to,

$$\sum_{i=1}^{N} \tau_{ij} = 1 \quad i = 1, 2, \cdots, N \tag{8}$$

$$\sum_{j=1}^{N} \tau_{ij} = 1 \quad j = 1, 2, \cdots, N \tag{9}$$

$$\sum_{k=1}^{N} k\tau_{ik} - \sum_{k=1}^{N} k\tau_{jk} \leq -1 \text{ if } \rho_{ij} - 1, \quad i, j = 1, 2, \cdots, N \tag{10}$$

$$\tau_{ij} \in \{0, 1\}, \quad i, j = 1, 2, \cdots, N \tag{11}$$

The objective function minimizes the average number of packet comparisons against policy list rules (7). Specifications (8) and (9) stipulate that every rule can only be present in the list once and that each list place can only include one rule (i.e., a one-to-one assignment). Constraint (10) enforces the rules' precedence relations. The decision variables are ultimately forced to be binary by constraints (11). Algorithm 1 shows the Probability-based sorting for ordering the firewall rules.

This algorithm calculates the rule weight using matching and precedence probability. If the precedence probability is less then or equal to 0, then the weight of the rule is also set as 0. Table 5 shows the weights for each rule.

Table 5. Rule weight

Rule ID	Weight-1	Weight-2
1	0.5	0.5
2	0.325	0.0
3	0.4857	0.0
4	1.1	1.1
5	0.52	0.0
6	0.75	0.75
7	0.5333	0.0
8	0.7999	0.7999
9	0.2	0.0
10	0.0	0.0

Algorithm 1 Probability-Based Sorting

Input:List of Firewall rules R=r_1,r_2,...,r_N
Output:Ordered Firewall Rules Index (Index-1 & Index-2)
for i = 1 *to* N **do**
Compute matching probability p_i using Eq. (1)
end
Find the binary decision variable τ_{ij} using Eq. (4)
Find precedence connection ρ_{ij} using Eq. (5)
for i = 1 *to* N **do**
Compute precedence probability Pm(i) using Eq. (6)
end
for i = 1 *to* N **do**
$w1_i = p_i + Pm(i)$
if *Pm(i) greater than 0* **then**
$w2_i = p_i + \mathrm{Pm}(i)$;
end
else
$w2_i = 0$
end
end
Sort w1 and w2 in descending order
OFRIndex-1 = sorted Index(w1)
OFRIndex-2 = sorted Index(w2)
Return OFRIndex-1 and OFRIndex-2

Table 6. Sorted rule index

Weight-1	r_4	r_8	r_6	r_7	r_5	r_1	r_3	r_2	r_9	r_{10}
Weight-2	r_4	r_8	r_6	r_1	r_2	r_3	r_5	r_7	r_9	r_{10}

Sort weight by descending order to get the rule order. The sorted rules are shown in Table 6. Apply the following formula to find the optimal rule ordering that minimizes:

$$\psi = \sum_{i=1}^{N} RI_i \omega_i \tag{12}$$

where RI_i represents the rule id and ω_i indicates the ith rule weight. The value of ψ for weight-1 is 25.93, and for weight-2 is 15.72. so the optimal rule list r_4, r_8, r_6, r_1, r_2, r_3, r_5, r_7, r_9, r_{10}. These sorted rule indexes are given to the input as the next approach (PSO) for rule optimization.

C. PSO-Based Optimized Rule Ordering

A population-based algorithm is Particle Swarm Optimization (PSO) [35]. Particle Swarm Optimization (PSO) is a population-based stochastic optimization technique inspired by the social behavior of birds flocking or fish schooling. It has several advantages compared to other optimization algorithms, such as Genetic Algorithms (GA), Simulated Annealing (SA), and traditional gradient-based methods. Here are some key advantages:

1) Simplicity and Ease of Implementation Simple Concept: PSO is conceptually simple and easy to understand. It involves fewer parameters and operations compared to algorithms like Genetic Algorithms (GAs), which require more complex operations like crossover and mutation. Fewer Parameters: PSO typically requires only a few parameters to be tuned, such as the number of particles, inertia weight, and learning factors. This contrasts with GAs, where parameters like population size, crossover rate, mutation rate, and selection methods must be carefully adjusted.
2) Efficient Global Search Global Optimization: PSO efficiently searches the global solution space by combining local and global exploration. The particles tend to converge to the global optimum by sharing information among them, reducing the chance of getting stuck in local minima. No Gradient Requirement: PSO does not require gradient information of the objective function, making it suitable for optimizing non-differentiable, discontinuous, or noisy objective functions.
3) Robustness Adaptability: PSO is robust to changes in the problem landscape, making it effective for dynamic optimization problems where the solution space may change over time. Multiple Solutions: PSO can explore multiple regions of the solution space simultaneously, increasing the likelihood of finding a global optimum in multi-modal optimization problems.
4) Convergence Characteristics Fast Convergence: PSO generally converges faster than GAs in many problems because it uses the best solution found so far to guide the search, whereas GAs may take longer due to the randomness involved in crossover and mutation. It functions by keeping a population or swarm of particles active. Each particle is a potential solution, represented by a vector of n real values, where n is the number of dimensions in the problem. In addition to position, every particle possesses a velocity, an n-dimensional vector whose dimensions indicate the direction and speed at which the particle should travel in the subsequent iteration. The particles are assessed using a fitness function for each iteration. To determine the optimal position for the entire population (gbest), the best position that each particle has previously investigated (pbest) is noted and shared across particles. A particle's velocity is then updated following its current velocity and these two optimal positions. Let X_i^t and V_i^t be the i-th particle's location and velocity vectors, respectively. The terms $\{pbest\}_i^t$ and $\{gbest\}_i^t$ represent the particle's personal best position and global best position, respectively, and are updated in the manner described below:

$$V_i^{t+1} = \omega V_i^t + c_1 r_1 \left(\text{ pbest } {}_i^t - X_i^t \right) + c_2 r_2 \left(\text{ gbest } {}_i^t - X_i^t \right) \tag{13}$$

$$X_i^{t+1} = X_i^t + V_i^{t+1} \tag{14}$$

where t is the iteration number, ω is the inertia weight, r_1 and r_2 are uniformly randomized between [0,1], and c1 and c2 are acceleration coefficients. PSO algorithm can be used in different domains. This work uses the PSO algorithm to optimize the firewall rule order. Algorithm 2 explains the proposed PSO algorithm for rule ordering.

Algorithm 2 PSO-based firewall rule optimization

Input:List of Firewall rules R=r_1,r_2,...,r_N
Output:Optimized firewall rule order for given rules
Initialize Parameters (maxIter, ω, c_1, c_2, r_1, r_2)
Initialize population
Initialize position and velocity
for *i = 1 to maxIter* **do**
for *p = 1 to number of population* **do**
Compute fitness values Fp using Eq. (12)
if *Fp is better than pbest* **then**
Update pbest;
end
end
end
Update gbest **for** *p = 1 to number of population* **do**
Update velocity using Eq. (13)
Update position using Eq. (14)
end
Return gbest (optimal firewall rule order)

The number of the population is set as 10. The initial first population of this algorithm is [4, 8, 6, 1, 2, 3, 5, 7, 9, 10] and the remaining are generated randomly. Table 7 shows the initial population of PSO. The fitness value of

Table 7. Initial population

r_4	r_8	r_6	r_1	r_2	r_3	r_5	r_7	r_9	r_{10}
r_4	r_8	r_6	r_7	r_5	r_1	r_3	r_2	r_9	r_{10}
r_8	r_1	r_2	r_3	r_{10}	r_7	r_5	r_9	r_6	r_4
r_3	r_5	r_7	r_1	r_2	r_4	r_9	r_8	r_6	r_{10}
r_5	r_6	r_9	r_1	r_3	r_2	r_4	r_8	r_9	r_{10}
r_9	r_1	r_3	r_2	r_4	r_6	r_{10}	r_7	r_5	r_8
r_{10}	r_4	r_7	r_3	r_1	r_2	r_5	r_6	r_8	r_9
r_4	r_5	r_6	r_7	r_9	r_1	r_2	r_8	r_{10}	r_3
r_3	r_2	r_4	r_1	r_{10}	r_6	r_8	r_9	r_5	r_7
r_5	r_2	r_6	r_3	r_1	r_7	r_8	r_9	r_{10}	r_4

each particle is computed using Eq. (12) and finds the pbest and gbest. The fitness value is 6.716, 1.901, 3.249, 2.153, 2.584, 2.090, 2.295, 2.677, 2.423, 2.878. The best optimal rule order is r_4, r_8, r_6, r_7, r_5, r_1, r_3, r_2, r_9, r_{10}.

V. Results and Discussion

The suggested optimal firewall rule ordering performance implementation details are examined in this section. The Java platform is used for its implementation, and cloudsim (a cloud simulator) creates and maintains the single Host and virtual machines. The system configuration comprises Windows 10 64-bit, an Intel Pentium 2.30 GHz processor, and 4 GB of RAM. These rules are Custom-created rules and arbitrary generated. The matching and precedence probability is computed for each rule and sorted based on the computed weights. Table 8 shows the Probability-based rule weight.

The rule order for probability based approach is r_{11}, r_3, r_4, r_8, r_2, r_1, r_5, r_6, r_7, r_9, r_{10}, r_{12}. Table 9 shows the population and fitness values. The optimal order of firewall rule is r_{11}, r_4, r_2, r_3, r_8, r_9, r_6, r_{12}, r_1, r_{10}, r_5, r_7. Table 10

Table 8. Probability-based rule weight

Rule ID	Weight-1	Weight-2
r_1	0.709	0.0
r_2	0.803	0.803
r_3	0.899	0.899
r_4	0.892	0.892
r_5	0.457	0.0
r_6	0.633	0.0
r_7	0.6	0.0
r_8	0.883	0.883
r_9	0.799	0.0
r_{10}	0.6	0.0
r_{11}	1.083	1.083
r_{12}	0.0	0.0

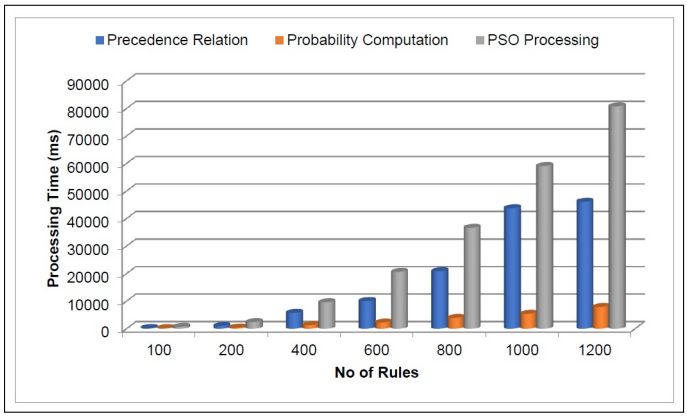

Fig. 2. Processing time comparison.

shows the execution time for precedence relation, probability computation and PSO processing time. Fig. 2 shows the execution time for different processes. When increasing the number of rules, the time also increases. The probability computation takes less time compared to other processes. The PSO takes longer to compare to precedence relation and probability computation. The performance of the proposed approach is influenced by the size of the population. The algorithm becomes stuck in local optima when there are insufficient particles, and it runs slowly when there are too many particles. There is no definitive guideline in literature for determining the optimum population size for cases, as the best size varies depending on the specific instance being solved. The substantial population size will amplify computing efforts and might impede rapid convergence. Table 11 shows the rule ordering time for different numbers of rules. Fig. 3 shows the comparison of rule ordering time. For 100 rules, 1.254sec and for 1000 rules, 108.349sec taken for rule ordering. When increasing the number of rules, the rule ordering time also increased.

Table 12 and Fig. 4 show the fitness value comparison for different rules. Table 13 shows comparison of different algorithms. Ordering firewall rules effectively is crucial for maximizing security and performance. Here are several applications where proper rule ordering can make a significant impact:

1) *Network Security*
Intrusion Prevention: By placing rules that block known malicious IP addresses or ports higher up, you

Table 9. Population and fitness value

Population	Fitness
$r_{11}, r_3, r_4, r_8, r_2, r_1, r_5, r_6, r_7, r_9, r_{10}, r_{12}$	8.306
$r_{11}, r_4, r_2, r_3, r_8, r_9, r_6, r_{12}, r_1, r_{10}, r_5, r_7$	2.275
$r_2, r_4, r_8, r_3, r_{11}, r_{10}, r_{12}, r_7, r_9, r_5, r_6, r_1$	2.676
$r_4, r_3, r_{11}, r_2, r_8, r_6, r_{10}, r_7, r_1, r_5, r_9, r_{12}$	2.500
$r_8, r_4, r_3, r_{11}, r_2, r_5, r_9, r_{10}, r_1, r_7, r_6, r_{12}$	2.645
$r_8, r_3, r_{11}, r_2, r_4, r_9, r_5, r_{10}, r_7, r_{12}, r_6, r_1$	3.195
$r_{11}, r_3, r_2, r_8, r_4, r_6, r_5, r_9, r_{10}, r_7, r_{12}, r_1$	2.903
$r_8, r_2, r_4, r_3, r_{11}, r_{10}, r_5, r_9, r_6, r_{12}, r_1, r_7$	3.115
$r_8, r_{11}, r_3, r_4, r_2, r_9, r_1, r_5, r_{10}, r_{12}, r_7, r_6$	2.970
$r_3, r_{11}, r_4, r_2, r_8, r_5, r_7, r_9, r_1, r_{10}, r_{12}, r_6$	2.615

Table 10. Execution time of different process

No of Rules	Precedence Relation Time (ms)	Probability Computation Time (ms)	PSO Processing Time(ms)
100	266	198	790
200	1047	494	2448
400	5745	1303	9612
600	10037	2298	20647
800	20892	3945	36736
1000	43804	5424	59121
1200	46202	7957	80820

can prevent threats from reaching sensitive areas of your network.

Access Control: Ensuring that rules allowing specific users or devices to access certain resources are prioritized helps enforce security policies effectively.

2) *Traffic Management*

Quality of Service (QoS): Prioritizing rules for critical applications (like VoIP or video conferencing) can help manage bandwidth and ensure these services have the necessary resources.

Load Balancing: Organizing rules to distribute incoming traffic evenly across multiple servers can enhance performance and reliability.

3) *Compliance and Auditing*

Regulatory Compliance: Ordering rules in alignment with compliance requirements (e.g., PCI-DSS, HIPAA) ensures that sensitive data is adequately protected.

Easier Audits: A well-ordered set of rules makes it simpler for auditors to review and verify that security measures are correctly implemented.

4) *Performance Optimization*

Faster Processing: By placing frequently matched rules at the top, you can reduce the time the firewall spends evaluating traffic, leading to improved overall network performance.

Resource Efficiency: Efficient rule ordering minimizes the processing burden on firewalls, allowing them to handle more connections simultaneously.

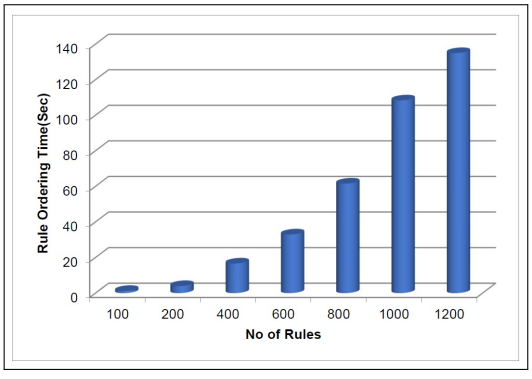

Fig. 3. Rule ordering time comparison.

Table 11. Rule ordering time

No of Rules	Rule Ordering Time (Sec)
100	1.254
200	3.989
400	16.660
600	32.982
800	61.573
1000	108.349
1200	134.979

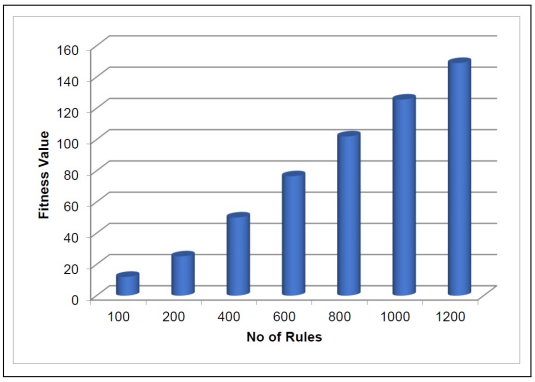

Fig. 4. Fitness value comparison.

The study revealed that the algorithm suggested in this work has the capability to rearrange more intricate scenarios.

Table 12. Fitness value comparison

No of Rules	Fitness Value
100	12.042
200	25.173
400	50.121
600	76.552
800	102.029
1000	125.616
1200	148.975

Table 13. Comparison with other Machine Learning Algorithms

No of Rules	Rule Ordering Time (Sec)		
	Simulated Annealing	Genetic	Proposed
100	8.269	5.305	1.254
200	12.31	7.923	3.989
400	27.528	20.351	16.660
600	43.093	37.25	32.982
800	79.128	70.42	61.573
1000	134.59	121.83	108.349
1200	162.42	150.39	134.979

With regards to the constraints of the suggested framework, it is worth noting that the parameters have been selected to enhance execution time while maintaining efficacy. This work only varies the size of population to find the processing time. The inertial weight can equilibrate the global and local search capabilities. A higher inertia weight is preferable for conducting global search, whereas a lower inertia weight is more ideal for performing local search. This work assigned a value of 0.1 to the inertia weight. The use of a linearly decreasing inertia weight can lead to improved performance in terms of processing time.

VI. Conclusion and Future Work

Firewall application screening is essential in situations involving fast network connectivity. In contemporary times, performance issues that hinder a service's ordinary operation can also worsen into significant cybersecurity concerns. Firewall policy optimization must be improved for these reasons. To minimize packet categorization latency, this research developed a dual technique that involves optimally rearranging firewall security rules. The proposed approach contains two algorithms; the first uses a probability-based approach for optimal firewall rule order, and the second uses Particle Swarm Optimization to find the complete solution. By examining the performance attained through an accurate optimization technique, the performance analysis of the suggested algorithm is expanded. The technique presented in this paper is more effective at rearranging more complex cases, as those in the Firewall rule class.The process of reordering can be intricate and susceptible to mistakes due to the extensive amount of rules and connections between rules. Deciding the most efficient sequence for rules can be intricate, particularly in settings with fluctuating or heavily congested circumstances. A potential direction for future work involves developing sophisticated simulation environments to evaluate the effectiveness of rule reordering strategies using a substantial number of rules and deep learning models.The future of firewall rule generation using probability method and then Particle Swarm Optimization presents numerous opportunities for enhancing network security. By focusing on automated, adaptive, and context-aware approaches, researchers and practitioners can develop more effective solutions that respond dynamically to evolving threats while maintaining performance and usability.

References

[1] A. Voronkov, L. A. Martucci, and S. Lindskog, "Measuring the usability of firewall rule sets," *IEEE Access*, vol. 8, pp. 27106–27121, 2020.

[2] T. Harada, K. Tanaka, R. Ogasawara, and K. Mikawa, "A rule reordering method via pairing dependent rules," pp. 1–9, 2020.

[3] M. Cheminod, L. Durante, L. Seno, and A. Valenzano, "Performance evaluation and modeling of an industrial application-layer firewall," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 5, pp. 2159–2170, 2018.

[4] U. F. . F. P. Alfredo De Santis, Aniello Castiglione, "An intelligent security architecture for distributed firewalling environments," *Journal of Ambient Intelligence and Humanized Computing*, vol. 4, pp. 223–234, 2013.

[5] S. Bagheri and A. Shameli-Sendi, "Dynamic firewall decomposition and composition in the cloud," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3526–3539, 2020.

[6] B. de Athayde Prata, L. R. Abreu, and J. Y. F. Lima, "Heuristic methods for the single-machine scheduling problem with periodical resource constraints," *TOP: An Official Journal of the Spanish Society of Statistics and Operations Research*, vol. 29, pp. 524–546, July 2021.

[7] S. Elloumi, T. Loukil, and P. Fortemps, "Reactive heuristics for disrupted multi-mode resource-constrained project scheduling problem," *Expert Systems with Applications*, vol. 167, p. 114132, 2021.

[8] X. Xu, H. Rong, M. Trovati, M. Liptrott, and N. Bessis, "Cs-pso: chaotic particle swarm optimization algorithm for solving combinatorial optimization problems," *Soft Computing*, vol. 22, pp. 783–795, 2016.

[9] A. H. Halim and I. Ismail, "Combinatorial optimization: comparison of heuristic algorithms in travelling salesman problem," *Archives of Computational Methods in Engineering*, vol. 26, pp. 367–380, 2019.

[10] S. Imran Hossain, M. Akhand, M. Shuvo, N. Siddique, and H. Adeli, "Optimization of university course scheduling problem using particle swarm optimization with selective search," *Expert Systems with Applications*, vol. 127, pp. 9–24, 2019.

[11] B. S. G. de Almeida and V. C. Leite, "Particle swarm optimization: A powerful technique for solving engineering problems," in *Swarm Intelligence* (J. D. Ser, E. Villar, and E. Osaba, eds.), ch. 3, Rijeka: IntechOpen, 2019.

[12] H. Hamed and E. Al-Shaer, "Dynamic rule-ordering optimization for high-speed firewall filtering," in *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security*, ASIACCS '06, (New York, NY, USA), pp. 332–342, Association for Computing Machinery, 2006.

[13] A. Tapdiya and E. W. Fulp, "Towards optimal firewall rule ordering utilizing directed acyclical graphs," in *2009 Proceedings of 18th International Conference on Computer Communications and Networks*, pp. 1–6, 2009.

[14] E. W. Fulp, "Optimization of network firewall policies using directed acyclical graphs (cid:0)," 2005.

[15] N. Ben Neji and A. Bouhoula, "Towards safe and optimal filtering rule reordering for complex packet filters," pp. 153–160, 2011.

[16] R. Mohan, A. Yazidi, B. Feng, and B. J. Oommen, "Dynamic ordering of firewall rules using a novel swapping window-based paradigm," in *Proceedings of the 6th International Conference on Communication and Network Security*, ICCNS '16, (New York, NY, USA), pp. 11–20, Association for Computing Machinery, 2016.

[17] R. Mohan, A. Yazidi, B. Feng, and J. Oommen, "On optimizing firewall performance in dynamic networks by invoking a novel swapping window–based paradigm," *International Journal of Communication Systems*, vol. 31, 2018.

[18] P. R. Kadam and V. K. Bhusari, "Redundancy removal of rules with reordering them to increase the firewall optimization," *International Journal of Research in Engineering and Technology*, vol. 3, pp. 317–321, 2014.

[19] T. Harada, K. Tanaka, and K. Mikawa, "Acceleration of packet classification via inclusive rules," in *Proc. 2018 IEEE Conference on Communications and Network Security (CNS)*, pp. 1–2, 2018.

[20] T. Harada, K. Tanaka, and K. Mikawa, "A heuristic algorithm for relaxed optimal rule ordering problem," in *Proc. 2018 2nd Cyber Security in Networking Conference (CSNet)*, pp. 1–8, 2018.

[21] T. Fuchino, T. Harada, K. Tanaka, and K. Mikawa, "Acceleration of packet classification using adjacency list of rules," in *Proc. 2019 28th International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–9, 2019.

[22] T. HARADA, K. TANAKA, and K. MIKAWA, "Simulated annealing method for relaxed optimal rule ordering," *IEICE Transactions on Information and Systems*, vol. E103.D, no. 3, pp. 509–515, 2020.

[23] A. Coscia, V. Dentamaro, S. Galantucci, D. Impedovo, and A. Maci, "A novel genetic algorithm approach for firewall policy optimization," in *Proc. 6th Italian Conference on Cybersecurity (ITASEC22)*, 06 2022.

[24] A. Coscia, V. Dentamaro, S. Galantucci, A. Maci, and G. Pirlo, "An innovative two-stage algorithm to optimize firewall rule ordering," *Computers and Security*, vol. 134, p. 103423, 08 2023.

[25] T. Fuchino, T. HARADA, K. Tanaka, and K. MIKAWA, "Computational complexity of allow rule ordering and its greedy algorithm," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E106.A, 03 2023.

[26] D. Bringhenti, L. Seno, and F. Valenza, "An optimized approach for assisted firewall anomaly resolution," *IEEE Access*, vol. PP, p. 1–1, 01 2023.

[27] B. Mor, D. Shabtay, and L. Yedidsion, "Heuristic algorithms for solving a set of np-hard single-machine scheduling problems with resource-dependent processing times," *Computers and Industrial Engineering*, vol. 153, p. 107024, 2021.
[28] S. Almufti, "Using swarm intelligence for solving nphard problems," *Academic Journal of Nawroz University*, vol. 6, pp. 46–50, 01 2017.
[29] R.-z. Zheng, Y. Zhang, and K. Yang, "A transfer learning-based particle swarm optimization algorithm for travelling salesman problem," *Journal of Computational Design and Engineering*, vol. 9, pp. 933–948, 05 2022.
[30] B. Wei, Y. Xing, X. Xia, and L. Gui, "An improved hybrid particle swarm optimization for travel salesman problem," in *Artificial Intelligence Algorithms and Applications* (K. Li, W. Li, H. Wang, and Y. Liu, eds.), (Singapore), pp. 514–525, Springer Singapore, 2020.
[31] L. Kou, J. Wan, H. Liu, W. Ke, H. Li, J. Chen, Z. Yu, and Q. Yuan, "Optimized design of patrol path for offshore wind farms based on genetic algorithm and particle swarm optimization with traveling salesman problem," *Concurrency and Computation: Practice and Experience*, vol. 36, no. 2, p. e7907, 2024.
[32] S. Alsaidy, A. Abbood, and M. Sahib, "Heuristic initialization of pso task scheduling algorithm in cloud computing," *Journal of King Saud University—Computer and Information Sciences*, vol. 34, 11 2020.
[33] M. Agarwal and G. M. S. Srivastava, "Opposition-based learning inspired particle swarm optimization (opso) scheme for task scheduling problem in cloud computing," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, pp. 1–21, 10 2021.
[34] P. Pirozmand, A. A. R. H. Hoda Jalalinejad, S. Mirkamali, and Y. Li, "An improved particle swarm optimization algorithm for task scheduling in cloud computing," *Journal of Ambient Intelligence and Humanized Computing*, vol. 14, pp. 4313–4327, 2023.
[35] M. Jain, V. Saihjpal, N. Singh, and S. B. Singh, "An overview of variants and advancements of pso algorithm," *Applied Sciences*, vol. 12, no. 17, 2022.
[36] F. Pizzato, D. Bringhenti, R. Sisto, and F. Valenza, "Automatic and optimized firewall reconfiguration," in *NOMS 2024-2024 IEEE Network Operations and Management Symposium*, pp. 1–9, 2024.
[37] C.-S. Chao, "Firewall rule optimization mechanism for ipv6-based iot networks," in *Proc. 2024 IEEE 4th International Conference on Electronic Communications, Internet of Things and Big Data (ICEIB)*, pp. 99–104, 04 2024.

Copyright © 2024 by the authors. This is an open-access article distributed under the Creative Commons Attribution License which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited (CC BY 4.0).