

Article

Centrality-Based Rule Ordering for Firewall Policy Optimization via Probability Propagation in Dependency Graphs

Fadwa Bezzazi * and Dounia Lotfi

Research Laboratory in Computer Science and Telecommunications (LRIT), Faculty of Sciences,
Mohammed V University in Rabat, B.P. 1014, Rabat 10000, Morocco; d.lotfi@um5r.ac.ma

* Correspondence: fadwa.bezzazi@um5r.ac.ma

Abstract

Firewall rule ordering aims to improve packet filtering efficiency while preserving the dependency constraints that guarantee the intended security behavior of the policy. Existing approaches often rely either on local criteria, such as rule frequency, or on iterative optimization procedures whose behavior depends on initialization, parameter settings and search budget. In this paper, we propose PPCO, a deterministic dependency-aware rule ordering method based on propagated probability combined with descendant-based centrality. The proposed score reflects both the traffic relevance of a rule and its structural influence in the dependency graph. The structural component is essential, especially when some rules are inactive or have zero activation probability, since it prevents probability-based ties from violating dependency constraints. The final policy is obtained directly by sorting rules in a decreasing score order. Experiments were conducted on synthetic rule sets ranging from 50 to 2000 rules and on ClassBench-ng benchmark instances, showing that PPCO consistently achieves a competitive ordering quality among the compared deterministic methods under the considered experimental settings. The method remains stable as the policy size and dependency rate increase, produces zero dependency violations in all valid configurations, achieves the lowest score-coherence values, and maintains competitive execution times at large scales. These results suggest that PPCO provides an effective, robust, and computationally efficient solution for dependency-aware firewall rule ordering within the scope of the evaluated configurations.

Keywords: firewall policy optimization; rule ordering; dependency graph; directed acyclic graph; packet classification; centrality; probability propagation; network security



Academic Editor: Christos Bouras

Received: 20 May 2026

Revised: 23 June 2026

Accepted: 30 June 2026

Published: 3 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Firewalls remain one of the main protection mechanisms in network security. They are used to control traffic flows, enforce access control policies, and prevent unauthorized communications between network entities. In practice, a firewall policy is represented as an ordered list of filtering rules, and packets are examined sequentially until the first matching rule is found [1,2]. This first-match behavior makes rule order a critical issue, since it directly affects both the applied security decision and the number of comparisons required to classify packets [3,4]. The firewall rule ordering problem aims to place the most important rules earlier in the policy while preserving the intended policy behavior. This problem is simple only when the rules are independent. In real firewall policies, dependencies often exist between rules because of overlap, exception patterns, and general-to-specific relations. In such cases, some rules must remain before others, and any invalid reordering

may modify the final filtering decision [2,5]. Firewall rule ordering is therefore not only a performance problem but is also a correctness problem constrained by precedence relations. From a broader optimization perspective, this type of constrained ordering is related to precedence-aware scheduling problems, which are known to be difficult in general [6]. Several methods have been proposed to improve firewall rule ordering. Some deterministic approaches rely on local criteria such as rule frequency or simple structural heuristics, while other approaches use iterative optimization techniques to explore the space of valid orderings [4,7–9]. These methods are useful, but they do not always reflect the full structural role of a rule inside a dependency-aware policy. A rule with moderate local activity may still be strategically important if it precedes a set of highly relevant descendant rules. In such a case, evaluating the rule only through its own frequency gives an incomplete view of its real importance. This observation is the starting point of the present work. Consider that the importance of a firewall rule should be evaluated through two essential attributes. The first is its activation probability, which reflects how often the rule is expected to be matched. The second is its structural influence in the dependency graph, which reflects how much of the descendant policy depends on its position. Based on these two attributes, we propose our method PPCO, a deterministic rule ordering method that combines propagated probability and descendant-based centrality in a single score. The propagated probability term extends the local view by including the contribution of all dependent descendants, while the centrality term captures the structural influence of the rule in the dependency graph. The final ordering is then obtained by sorting rules in decreasing score order. The originality of the method lies in this dependency-aware interpretation of rule importance. PPCO does not rely on iterative search, random initialization, or budget-dependent refinement. Instead, it produces a direct ranking that remains simple, interpretable, and structurally grounded. This makes the method robust from two perspectives. It remains compatible with the precedence constraints that preserve the security semantics of the policy, and it also remains computationally light when the policy size grows. In this sense, the proposed approach extends our previous work on firewall rule ordering with Discrete Cuckoo Search by moving from stochastic search-based optimization to a deterministic dependency-aware ranking strategy [10]. The proposed method is evaluated on both synthetic and benchmark rule sets. The synthetic policies range from 50 to 2000 rules and are used to study the effect of policy size, dependency rate, and score parameters under controlled settings. In addition, ClassBench-ng rule sets are used to validate the method on benchmark instances closer to practical packet classification structures. The results show that PPCO consistently produces valid orderings, achieves the lowest score-coherence values among the compared methods, and remains competitive in fitness and execution time as the policy size and dependency density increase. The remainder of this paper is organized as follows. Section 2 presents the firewall background and the formal problem definition. Section 3 reviews the related work. Section 4 describes the proposed methodology. Section 5 reports the experimental evaluation and discusses the results. Section 6 concludes the paper and outlines future directions.

2. Firewall Background

Firewalls remain among the most widely used mechanisms for protection in computer networks. Their aim is to control traffic flows and enforce access control policies between different network zones. In practice, a firewall inspects packet header fields such as source address, destination address, source port, destination port, and protocol, and then applies the action associated with the first rule that matches the packet [1,2]. This filtering process makes the firewall a central component for protecting internal resources, restricting unauthorized access, and regulating communications according to the security requirements

of the system. A firewall policy is therefore not only a list of filtering conditions; it is an operational representation of the security policy of the network. For this reason, two aspects must be respected simultaneously: First is the correctness, since the policy must enforce the intended security decisions without ambiguity. Second, is efficiency, since the policy should classify packets with the minimum comparison operations. In first-match firewalls, these two aspects are directly linked to the order of rules [3,4]. Considering a packet that arrives, the firewall starts comparing it sequentially with the rules of the policy until a match is found. The action of this first matching rule is then applied. As a result, placing rules with a high frequency rate near the top of the policy reduces the average packet matching cost automatically. However, rule ordering cannot be modified freely in all scenarios. Some rules overlap in their matching space, and their relative order determines which action will be applied to specific packets. This leads us to say that reordering rules may improve performance, but it may also change the intended behavior of the policy if precedence relations are not preserved [11,12]. To illustrate this point, we consider the simplified firewall policy shown in Table 1, which r_1 allows HTTP traffic from the internal network to a specific web server, while rule r_2 denies all traffic from the same internal network to a larger destination subnet. If r_1 is placed before r_2 , HTTP access to the web server remains allowed as intended. If the order is reversed, the broader deny rule may match the packet first, and the specific allow rule becomes ineffective. This simple example explains that two individually valid rules may produce different filtering outcomes depending on their order [2,5].

Table 1. Illustrative firewall policy showing the importance of rule order. The symbol * denotes any value.

Rule	Source IP	Destination IP	Destination Port	Protocol	Action
r_1	10.0.0.0/8	172.16.1.10/32	80	TCP	Allow
r_2	10.0.0.0/8	172.16.1.0/24	*	*	Deny
r_3	*	*	*	*	Deny

This example reflects a common situation in firewall administration, where specific exception rules coexist with broader filtering rules. In such cases, the policy must satisfy several security criteria. A valid ordering must preserve the intended semantics of access control, avoid contradictory interpretations, and maintain deterministic filtering behavior. At the same time, it should reduce unnecessary packet comparisons whenever possible. Firewall policy management is therefore not limited to defining correct rules, but it also requires maintaining a consistent and efficient rule sequence [5,11,12]. These considerations explain why firewall rule ordering is a constrained problem rather than a simple sorting task. Some rules can be moved safely, while others cannot because they participate in dependency relations. A correct reordering must preserve the precedence constraints induced by these dependencies, and this is why dependency-aware models are needed when optimizing firewall policies. In the present work, the dependency structure is represented explicitly and used as the basis for rule ranking and final ordering.

3. Related Work

Firewall rule ordering has been studied from different perspectives in the literature. A first line of work focuses on the correctness of firewall policies, especially the detection of anomalies such as shadowing, redundancy, conflict, and correlation between rules. These studies showed that the behavior of a firewall policy cannot be understood by examining each rule independently, because the relative position of rules directly affects the final filtering decision. Early works by Al-Shaer and Hamed, and later by Abedin et al., highlighted

the importance of analyzing interactions between rules in order to preserve policy consistency and avoid unintended decisions [11–13]. Their work established an essential point for later ordering methods: reordering rules is only meaningful if the security semantics of the policy remain unchanged. A second line of research addresses the performance side of the problem. In first-match firewalls, the order of rules affects the average number of comparisons needed to classify packets. Several deterministic approaches have therefore been proposed to improve packet filtering efficiency while preserving correctness constraints. Fulp modeled rule dependencies through directed acyclic graphs and showed that graph-based representations are useful for identifying admissible reorderings [2]. Dobrescu and Argyraki also studied dynamic rule ordering for high-speed filtering, with the objective of reducing processing cost under realistic traffic conditions [3]. In the same direction, El-Alfy proposed a heuristic but constructive approach for firewall policy optimization that aims to improve ordering quality without violating rule interactions [4]. Another important contribution in this area is the work of Ben Neji and Bouhoula, who addressed safe and optimal rule reordering for complex packet filters [5]. Their study emphasized that safe reordering requires an explicit representation of rule relations, and it introduced a matrix-based view of filtering rule interactions. This perspective is particularly relevant to the present work because it connects the feasibility of reordering with the structure of inter-rule dependencies. It also supports the idea that the importance of a rule should not be reduced to its local matching behavior only, but should be interpreted in relation to the policy structure. In the context of cloud-based deployments, Lu and Yang addressed firewall rule ordering for embedded network firewalls handling a large number of rules [14]. Their work analyzed the different types of conflicts that can occur between firewall rules and proposed a rule order adjustment method designed to dynamically update rule priorities while preserving the original semantics of the policy. This confirms that the rule ordering problem remains relevant in modern cloud environments, where the number of rules and the frequency of policy updates are typically higher than in traditional on-premise deployments, and it extends naturally to next-generation and zero-trust firewall architectures, where fine-grained, frequently updated rule sets play a similarly central role in enforcing access control policies. More recently, Fonseca et al. proposed a two-stage method for firewall rule ordering that combines a structural analysis phase with a frequency-based placement phase [15]. Their method first divides rules into independent and dependent groups, and then applies a greedy local criterion within each group to minimize the expected matching cost. This approach confirms that structured rule analysis improves policy organization and filtering efficiency. However, it evaluates each rule according to its local frequency only, without propagating importance through the dependency chain. As a result, a rule with moderate direct activity but many active descendants may be undervalued by this method. More recently, Coscia et al. proposed FROG, a firewall rule order generator that produces constraint-compliant orderings and arranges rules into jumpable blocks based on their structural similarity, without requiring prior knowledge of traffic distributions [16]. While FROG targets block-skipping firewalls and focuses on minimizing packet-rule comparisons through structural clustering, PPCO differs by assigning each rule a global dependency-aware score that combines propagated activation probability with descendant-based centrality, enabling a direct and interpretable ranking without iterative refinement. The proposed PPCO method addresses this limitation explicitly through its propagated probability term. The methods selected as baselines in this paper are those whose algorithmic behavior could be reproduced precisely under the same experimental conditions. In parallel with these deterministic and graph-based approaches, several metaheuristic methods have been applied to firewall rule ordering. The motivation behind these methods is that the problem becomes difficult when the search

space is large and constrained. General optimization techniques such as simulated annealing, genetic algorithms, and random search have long been used for hard combinatorial problems [7–9]. Their application to firewall rule ordering follows the same logic: instead of building the final order directly, they explore a large set of candidate permutations and attempt to improve them iteratively according to an objective function. Our previous work belongs to this second category. In that study, we proposed a Discrete Cuckoo Search method for firewall rule ordering and showed that it can produce competitive results under dependency constraints [10]. The cuckoo search family, originally introduced by Yang and Deb and later extended to discrete combinatorial settings, is attractive because it combines global exploration and local refinement [17–19]. Such methods are useful when the ordering problem is treated as a constrained optimization task over a large permutation space. They can provide good solutions, especially when constructive rules are difficult to define. However, they also have limitations. Their performance often depends on several control parameters, on the initialization strategy, and on the number of iterations allowed during the search. In addition, their behavior may vary from one run to another, even on the same policy instance. Beyond firewall rule ordering specifically, probabilistic graph-based modeling has also been applied to other cybersecurity problems, such as the fuzzy GERT model proposed by Semenov et al. for investigating software vulnerabilities under uncertainty [20], illustrating a broader interest in combining graph structures with probabilistic reasoning across the field. The method proposed in this paper is positioned at the intersection of these three research directions. From the correctness literature, it retains the requirement that any valid reordering must preserve the semantic constraints induced by rule dependencies. From deterministic and graph-based methods, it retains the structural view of the policy and the objective of producing a safe and interpretable rule order without iterative search. From the metaheuristic literature, it retains the idea that rule importance should be evaluated globally rather than locally, and that the quality of the final order depends on how well the method exploits the structure of the problem. With respect to existing deterministic methods, PPCO differs from frequency-based approaches such as Greedy by considering not only the local activation probability of each rule but also the cumulative contribution of all its descendants in the dependency graph. It differs from the two-stage method of Fonseca et al. in that it does not partition rules into groups but assigns a unified dependency-aware score to each rule and derives the final ordering by a single sort. With respect to metaheuristic approaches, PPCO avoids stochastic behavior and parameter sensitivity by construction, while still capturing a non-local notion of rule importance through probability propagation. To the best of our knowledge, the combination of propagated probability and descendant-based centrality into a single deterministic score has not been explicitly formulated in previous firewall rule ordering methods. This combination constitutes the main originality of the proposed approach.

4. Methodology

4.1. Problem Statement

Let $R = \{r_1, r_2, \dots, r_n\}$ be a firewall policy composed of n ordered filtering rules. Each rule is defined over the standard five-tuple fields, namely source IP address, destination IP address, source port, destination port, and protocol, and is associated with a filtering decision such as `allow` or `deny`. In a first-match firewall, packets are examined sequentially, and the action of the first matching rule is applied [1,2]. This means that the position of a rule in the policy directly affects both filtering efficiency and the final security decision.

The objective addressed in this work is not to search iteratively for an optimal permutation, but to construct a valid rule order that places the most important rules earlier in the policy while preserving all precedence constraints induced by dependencies. In the

absence of dependencies, a natural ordering strategy is to place the most probable rules at the beginning of the list, since highly active rules contribute more to the average filtering behavior [3,4]. However, real firewall policies are not composed of independent rules. They often contain semantic dependencies caused by overlap, exception patterns, or general-to-specific relations [5,11,12]. In such cases, some rules must remain before others, and any invalid reordering may change the intended behavior of the policy.

The rule ordering problem considered in this paper can therefore be stated as follows. Given a set of firewall rules R , a dependency structure over R , and an activation probability $P(r_i)$ associated with each rule r_i , the objective is to build a final ordering π such that two conditions are satisfied. First, the ordering must preserve all dependency constraints. Second, it must place the most influential rules as early as possible in the policy. In this context, the influence of a rule is not reduced to its local activation probability. It also depends on the descendant rules whose validity or effectiveness is structurally related to it. The problem is therefore both probabilistic and structural.

Based on this view, the proposed method does not rank rules only according to their local activity. Instead, it assigns each rule a dependency-aware score that combines local probability, propagated descendant probability, and structural influence in the dependency graph. The final policy is then obtained by sorting rules in decreasing order of this score. The role of the methodology is therefore to define a score that is meaningful from a traffic perspective, consistent with the dependency structure, and sufficient to produce a valid deterministic ordering.

4.2. Graph Modeling of Rule Dependencies

To represent precedence constraints explicitly, the firewall policy is modeled as a directed acyclic graph $G = (R, E)$, where the vertices correspond to the firewall rules and the edges represent dependency relations [2,21]. A directed edge

$$(r_i, r_j) \in E$$

means that rule r_i must appear before rule r_j in any valid ordering. Under this convention, r_i is a predecessor of r_j , and r_j is a successor of r_i . Since these relations define a partial order, the graph must be acyclic. A cycle would imply contradictory precedence requirements and would make the policy impossible to reorder consistently [6,21].

For each rule r_i , the set of all descendants of r_i is denoted by $Desc(r_i)$ and defined as

$$Desc(r_i) = \{r_j \in R \mid \text{there exists a directed path from } r_i \text{ to } r_j\}.$$

This transitive definition is central to the present work. The influence of a rule is not limited to its immediate successors. A rule may control an entire dependency chain, and this structural effect should be reflected in the ranking process. This graph-based interpretation is consistent with previous matrix- and DAG-based studies on safe firewall rule reordering [2,5].

The same dependency information can also be represented in matrix form through a binary dependency matrix $D \in \{0, 1\}^{n \times n}$, where

$$D_{ij} = \begin{cases} 1 & \text{if } r_i \text{ must precede } r_j, \\ 0 & \text{otherwise.} \end{cases}$$

The graph representation is useful to visualize the structure of the policy, while the matrix representation is convenient for computation, verification, and experimental analysis. In the proposed method, the graph is used to identify descendant sets and quantify the

structural role of each rule, whereas the matrix offers a compact representation of precedence constraints. This modeling choice also explains why local probability alone is not sufficient to characterize rule importance. A rule may have moderate direct activity and still be critical because several important rules depend on it. In such a case, its role in the policy extends beyond its own matching rate. This motivates the use of a score that combines traffic information with dependency-based structural information.

4.3. PPCO Score Formulation

The proposed method assigns each rule a score that combines two complementary aspects of importance. The first is probabilistic and reflects the activity associated with the rule. The second is structural and reflects its position and influence in the dependency graph. The score of rule r_i is defined as

$$\text{Score}(r_i) = \alpha \left(P(r_i) + \sum_{r_j \in \text{Desc}(r_i)} P(r_j) \right) + \beta \frac{|\text{Desc}(r_i)|}{n-1},$$

where $P(r_i)$ is the local activation probability of rule r_i , $\text{Desc}(r_i)$ is the set of all descendants of r_i , n is the total number of rules, and $\alpha, \beta \geq 0$ are weighting coefficients such that $\alpha + \beta = 1$. The first term corresponds to the propagated probability. It combines the local probability of the rule with the cumulative probabilities of all rules that depend on it, directly or indirectly. This term captures the idea that a rule is important not only because it may be matched frequently, but also because it may condition the position and relevance of descendant rules. A predecessor that governs a chain of highly active descendants should therefore receive a higher priority than a rule considered only in isolation. This extends the local view traditionally used in rule ordering and remains consistent with dependency-aware interpretations proposed in earlier studies of safe reordering [5]. The second term corresponds to a normalized structural centrality. It is inspired by the out-degree view of influence, but it is extended here to the full descendant set rather than only direct successors. This choice is more appropriate for dependency-aware firewall ordering because the structural effect of a rule may propagate over several levels of the graph. A rule that precedes many descendant rules occupies a stronger structural position than a terminal rule with no descendants. In this sense, the centrality term captures the role of a rule as a structural gate in the policy. The coefficients α and β control the balance between these two components. The parameter α weights the propagated probability term, while β weights the structural centrality term. A larger value of α gives more importance to traffic behavior, whereas a larger value of β emphasizes structural influence. The purpose of these coefficients is not to create an arbitrary linear combination, but to balance two distinct dimensions of rule importance. The probabilistic term remains the main driver of the ranking, while the structural term acts as a stabilizing factor when local probabilities are not sufficient to discriminate safely between rules. The choice of the weighting coefficients was examined experimentally through a systematic evaluation over the range $\alpha \in \{0.1, 0.2, \dots, 1.0\}$ with $\beta = 1 - \alpha$. The special case $\alpha = 1$ and $\beta = 0$ corresponds to an ordering based only on the propagated activation probability. Although this configuration may provide a slightly higher raw fitness value, it is not considered admissible because it removes the structural component of the score. In practical firewall policies, some rules may be inactive during the observation period and may therefore have zero activation probability. When such inactive rules are involved in dependency relations, the probability-based component alone may fail to impose a strict order between a predecessor and its dependent successor. This can create ties or non-discriminative rankings, allowing a sorting procedure to place dependent rules in an invalid order. The descendant-based centrality term addresses this limitation.

Even when the activation probability of a predecessor is zero, the predecessor has a larger descendant set than its dependent successor. As a result, any non-zero value of β gives a strictly positive structural advantage to the predecessor. The role of β is therefore not only to improve the ranking quality, but also to preserve feasibility when traffic information is incomplete, sparse, or non-discriminative. For this reason, only configurations with $\beta > 0$ are considered admissible in the proposed PPCO method. For all valid configurations with $\alpha < 1$, the variation in fitness remained marginal, with a difference below 0.04% between $\alpha = 0.6$ and $\alpha = 0.9$. This indicates that the probabilistic component remains dominant across the valid range, regardless of the exact balance. The value $\alpha = 0.7$ was therefore retained because it achieved a competitive valid fitness while preserving a non-zero structural contribution $\beta = 0.3$, which is sufficient to resolve ties when probabilities are uninformative. This choice is also consistent with the theoretical property established below, which requires only $\beta > 0$ to preserve the dependency order. The motivation of the formulation is therefore straightforward. A purely local frequency-based ranking is effective only when probabilities are clearly discriminative and dependencies do not create ambiguity. In realistic dependency-aware policies, the relevance of a rule also depends on the rules that rely on it. The proposed score addresses this limitation by combining propagated traffic importance with dependency-based structural influence. The final policy is then obtained by sorting rules in decreasing score order.

4.4. Theoretical Property of the Score

An important property of the proposed score is that it is consistent with the dependency structure. More precisely, if a rule must precede another rule, then its score is strictly higher. This property explains why sorting the rules in decreasing score order preserves all dependency constraints.

Proposition 1. Let $G = (R, E)$ be a dependency DAG. For any dependency edge $r_i \rightarrow r_j$, if $P(r) \geq 0$ for every rule $r \in R$, $\alpha \geq 0$, and $\beta > 0$, then

$$\text{Score}(r_i) > \text{Score}(r_j).$$

Proof. Since $r_i \rightarrow r_j$, rule r_j is a descendant of r_i . Moreover, because the graph is acyclic, every descendant of r_j is also a descendant of r_i . Therefore, there exists a set E_{ij} , possibly empty, such that

$$\text{Desc}(r_i) = \text{Desc}(r_j) \cup \{r_j\} \cup E_{ij}.$$

Using the score definition, we obtain

$$\begin{aligned} \text{Score}(r_i) - \text{Score}(r_j) = & \alpha \left(P(r_i) + \sum_{d \in \text{Desc}(r_i)} P(d) - P(r_j) - \sum_{d \in \text{Desc}(r_j)} P(d) \right) \\ & + \beta \left(\frac{|\text{Desc}(r_i)| - |\text{Desc}(r_j)|}{n - 1} \right). \end{aligned}$$

By substituting the decomposition of $\text{Desc}(r_i)$, the probability term becomes

$$P(r_i) + \sum_{e \in E_{ij}} P(e),$$

and the structural term becomes

$$|\text{Desc}(r_i)| - |\text{Desc}(r_j)| = 1 + |E_{ij}|.$$

Thus,

$$\text{Score}(r_i) - \text{Score}(r_j) = \alpha \left(P(r_i) + \sum_{e \in E_{ij}} P(e) \right) + \beta \frac{1 + |E_{ij}|}{n - 1}.$$

Since $P(r) \geq 0$ for every rule r , the first term is non-negative. Since $\beta > 0$, $n > 1$, and $1 + |E_{ij}| > 0$, the second term is strictly positive. Therefore,

$$\text{Score}(r_i) - \text{Score}(r_j) > 0.$$

Hence,

$$\text{Score}(r_i) > \text{Score}(r_j).$$

□

This proves that every predecessor receives a strictly higher score than its dependent successor, even when some activation probabilities are equal to zero.

Corollary 1. *Sorting the rules in decreasing order of $\text{Score}(r_i)$ yields an ordering that satisfies all dependency constraints. This follows directly from Proposition 1, in particular $\beta > 0$. If every predecessor has a strictly higher score than each of its dependent successors, then a global descending sort cannot place a successor before one of its predecessors. The score therefore acts not only as an importance measure, but also as an ordering mechanism compatible with the dependency graph. This gives the proposed method a deterministic and interpretable foundation.*

4.5. Illustrative Example

To illustrate the proposed method, consider the small firewall policy of six rules shown in Table 2. The example shows each step of the method, from dependency modeling to score computation and final ranking.

Table 2. Illustrative firewall policy with six rules. The symbol * denotes any value.

Rule	Source IP	Destination IP	Destination Port	Protocol	Action
r_1	10.0.0.0/8	172.16.0.0/16	*	*	Deny
r_2	10.1.0.0/16	172.16.1.0/24	80	TCP	Allow
r_3	10.2.0.0/16	172.16.2.0/24	443	TCP	Allow
r_4	10.1.1.0/24	172.16.1.10/32	80	TCP	Deny
r_5	10.1.2.0/24	172.16.1.20/32	80	TCP	Deny
r_6	10.2.1.0/24	172.16.2.10/32	443	TCP	Deny

The dependency relations induced by this policy are represented by the DAG shown in Figure 1. In this example, rule r_1 is a general predecessor of both r_2 and r_3 . Rule r_2 precedes the more specific rules r_4 and r_5 , while rule r_3 precedes rule r_6 . This structure reflects a common pattern in firewall policies, where general rules are followed by more specific exceptions [2,5].

The same structure can be written in matrix form as shown in Table 3. This matrix represents the direct dependency relations between rules.

Let the activation probabilities of the six rules be those given in Table 4. They are chosen here only for illustration and satisfy the normalization condition.

From the graph, the descendant sets are

$$\text{Desc}(r_1) = \{r_2, r_3, r_4, r_5, r_6\}, \quad \text{Desc}(r_2) = \{r_4, r_5\}, \quad \text{Desc}(r_3) = \{r_6\},$$

$$\text{Desc}(r_4) = \emptyset, \quad \text{Desc}(r_5) = \emptyset, \quad \text{Desc}(r_6) = \emptyset.$$

Using $\alpha = 0.7$ and $\beta = 0.3$, the scores are computed as shown in Table 5.

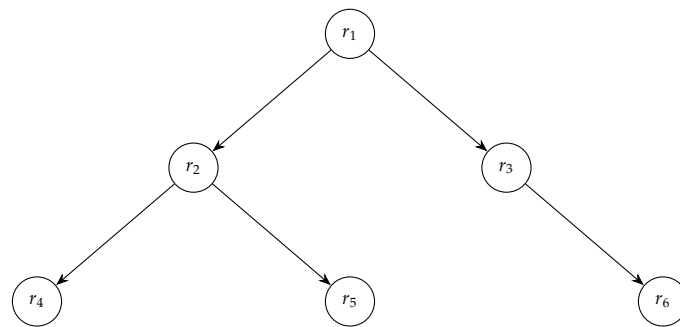


Figure 1. Dependency DAG of the illustrative firewall policy. An edge $r_i \rightarrow r_j$ means that r_i must precede r_j .

Table 3. Direct dependency matrix of the illustrative example.

D	r_1	r_2	r_3	r_4	r_5	r_6
r_1	0	1	1	0	0	0
r_2	0	0	0	1	1	0
r_3	0	0	0	0	0	1
r_4	0	0	0	0	0	0
r_5	0	0	0	0	0	0
r_6	0	0	0	0	0	0

Table 4. Activation probabilities of the illustrative example.

Rule	r_1	r_2	r_3	r_4	r_5	r_6
$P(r_i)$	0.08	0.17	0.12	0.23	0.18	0.22

Table 5. Score computation for the illustrative example.

Rule	Propagated Probability	Centrality	Score	Rank
r_1	1.00	1.00	1.000	1
r_2	0.58	0.40	0.526	2
r_3	0.34	0.20	0.298	3
r_4	0.23	0	0.161	4
r_6	0.22	0	0.154	5
r_5	0.18	0	0.126	6

The final ordering obtained by the proposed method is therefore $r_1 \succ r_2 \succ r_3 \succ r_4 \succ r_6 \succ r_5$. This ordering is valid because every predecessor appears before its dependent successors. At the same time, the ranking reflects the combined influence of probability and graph structure. Rule r_1 is placed first not because of its local probability, which is relatively low, but because it dominates the whole dependency structure. Rule r_2 is ranked above r_3 because it controls two descendants instead of one and accumulates a larger propagated probability. The terminal rules r_4 , r_5 , and r_6 are ranked only according to their local probabilities, since they have no descendants. This example shows the main behavior of the method. The score gives priority to rules that are both active and structurally influential. As a result, the final ordering remains consistent with the dependency graph while going beyond a purely local frequency-based ranking.

4.6. Computational Complexity

The computational complexity of PPCO can be analyzed by decomposing the method into two phases. Given a policy of n rules and a dependency graph $G = (R, E)$ with $|E|$ edges, the method proceeds as follows. The first phase computes the descendant set

$\text{Desc}(r_i)$ and the propagated probability for each rule. In the reference implementation, this is performed by an independent traversal of the DAG from every rule, computing the full set of transitive descendants of r_i before accumulating their probabilities and evaluating the centrality term. A single traversal from one rule runs in $O(n + |E|)$, and since this traversal is repeated independently for every one of the n rules, the first phase runs in $O(n(n + |E|))$. The second phase sorts the n rules in decreasing score order, which requires $O(n \log n)$ time using a standard comparison-based sort. Since the dependency rate is bounded by the maximum out-degree constraint (fixed to 3 in all experiments), the number of edges satisfies $|E| \leq 3n$, so the first phase dominates the overall cost. The total complexity of PPCO is:

$$O(n(n + |E|)) + O(n \log n) = O(n^2).$$

The three baseline methods follow the same general pattern: a score or weight-computation phase based on per rule reachability information, followed by an iterative selection procedure, as shown in Table 6.

Table 6. Algorithmic complexity of the four methods.

Method	Score Computation	Ordering Step
PPCO (ours)	$O(n(n + E))$	$O(n \log n)$
Greedy	$O(1)$ per rule	$O(n^2)$
SGM	$O(n(n + E))$	$O(n^2)$
ZHTO-D1	$O(n(n + E))$	$O(n^2)$

The score computation complexity of PPCO reflects the same asymptotic score-computation cost $O(n(n + |E|))$, since all three require, for each rule, a traversal of its reachability set in the dependency graph (descendants for PPCO, ancestors for SGM and ZHTO-D1). The methods differ, however, in their ordering step. Greedy, SGM, and ZHTO-D1 build the final ordering through an iterative kahn-style selection procedure: at each of the n steps, the currently admissible rules are scanned to identify the best candidate, which costs $O(n)$ per step and therefore $O(n^2)$ overall. PPCO replaces this iterative selection with a single global sort by score, which costs $O(n \log n)$. The overall worst-case complexity of all four methods is therefore $O(n^2)$, dominated in every case by the score or weight-computation phase rather than by the ordering step. This asymptotic equivalence is consistent with the empirical execution times reported later in Section 5. At the smallest tested size ($n = 50$), Greedy is marginally faster than PPCO, since its constant-time local selection criterion incurs negligible overhead when the rule set is small. From $n = 100$ onwards, PPCO becomes consistently faster than all three baselines, and the gap widens with n : the speedup over Greedy grows from $1.2\times$ at $n = 100$ to $15.9\times$ at $n = 2000$. This growing advantage likely reflects a smaller practical constant in PPCO's ordering phase, which avoids the repeated re-scanning of the admissible rule set performed by Greedy, SGM and ZHTO-D1 at every step of their Kahn-style construction.

5. Experimental Evaluation

5.1. Experimental Setup

using the NetworkX library for graph operations and NumPy for numerical computations. Execution times were additionally verified on Google Colab for the largest synthetic rule sets up to 2000 rules in order to confirm the runtime behavior at larger scales. The first category of datasets consists of synthetic firewall rule sets generated under a controlled dependency model. Each rule is represented as a five-tuple over IP prefixes and port ranges drawn from a fixed hierarchy. Dependency relations are identified through semantic subsumption, and the resulting precedence structure is represented as a DAG.

Rule set sizes range from 50 to 2000 rules. The dependency rate was varied from 0% to 100%, and the maximum out-degree per node was fixed to 3 in all synthetic experiments. Each configuration was evaluated over 10 independent random seeds, and average values are reported. The second category consists of benchmark rule sets generated with ClassBench-ng [22] using the fw1 seed file, which statistically models enterprise firewall policies. Due to the internal filtering behavior of the generator, the obtained rule sets do not always match the requested sizes exactly. In our experiments, the resulting benchmark instances contain 50, 88, 122, and 240 rules. The proposed method, PPCO, is compared with three baseline ordering strategies. Greedy is a dependency-preserving local strategy that selects, at each step, the currently admissible rule with the highest local activation probability. SGM is a graph-based baseline in which each rule is assigned a weight derived from the mean probability of its reachable predecessor set, and admissible rules are selected according to this value. ZHTO-D1 is the first deterministic stage of the two-stage method proposed by Coscia et al. [15]. It prioritizes rules according to a threshold-based frequency split combined with a graph-based reachability criterion. The second stage of the original method relies on a genetic algorithm with stochastic initialization, which is outside the scope of this deterministic comparison and was therefore not reproduced. In the remainder of this paper, ZHTO-D1 refers exclusively to this deterministic first stage. PPCO differs from these baselines by assigning each rule a global dependency-aware score and obtaining the final policy through a direct descending sort. The three baseline methods were reimplemented from scratch based on their original algorithmic descriptions, since no reference implementation is publicly available. For ZHTO-D1, only the first deterministic stage of the method proposed by Coscia et al. [15] was reproduced, as the second stage relies on a genetic algorithm whose behavior depends on stochastic initialization and is therefore outside the scope of this deterministic comparison.

5.2. Evaluation Metrics

Four evaluation metrics are used throughout the experiments. The first metric is the ordering fitness

$$F(\sigma) = \sum_{i=1}^n P(r_{\sigma(i)}) (n - i + 1),$$

where $r_{\sigma(i)}$ is the rule placed at position i in the final ordering, $P(r_{\sigma(i)})$ is its activation probability, and n is the number of rules. This metric gives the highest contribution to highly probable rules placed near the beginning of the policy. A higher value of $F(\sigma)$ therefore indicates a better ordering. The second metric is the Score-Coherence Fitness (SCF), defined as

$$\text{SCF}(\sigma) = \sum_{i=1}^n \text{Score}(r_{\sigma(i)}) \cdot i.$$

This metric evaluates how well the final ordering is aligned with the proposed score. Lower SCF values indicate that high-score rules are placed earlier, which is the intended behavior of PPCO. The third metric is the number of dependency violations, defined as the number of precedence relations $r_i \rightarrow r_j$ such that r_j appears before r_i in the final ordering. A valid ordering must produce zero violations. The fourth metric is the execution time in seconds, used to assess the computational efficiency of each method. In addition to these four metrics, Section 5.3 introduces a fifth metric, the ratio to the exact optimal fitness $F(\sigma)/F(\sigma^*)$, which is used only for the small-scale validation against the true optimum, since computing $F(\sigma^*)$ is only tractable for small rule sets.

5.3. Results

Activation probabilities follow a Zipf distribution with parameter $a = 1.2$ [23], normalized such that $\sum_i P(r_i) = 1$. The weighting coefficients of PPCO were evaluated over the range $\alpha \in \{0.1, 0.2, \dots, 1.0\}$ with $\beta = 1 - \alpha$. The configuration $\alpha = 0.7, \beta = 0.3$ was retained because it provided a good fitness value while preserving zero violations.

5.3.1. Influence of the Weighting Coefficients

Table 7 reports the effect of the coefficients α and β on PPCO for $n = 200$ rules and a dependency rate of 30%. As α increases, the fitness improves steadily reaching its highest value among all valid configurations at $\alpha = 0.9, \beta = 0.1$. The configurations $\alpha = 1, \beta = 0$, which removes the structural term completely, achieve a slightly higher raw fitness but produce dependency violations. This confirms that local and propagated probability alone are not sufficient to guarantee safe ordering in all cases. Although $\alpha = 0.9$ yields the highest fitness among the valid settings, the difference with $\alpha = 0.7$ is marginal, while a smaller β leaves a thinner structural margin to resolve ties when probabilities are weakly discriminative. The setting $\alpha = 0.7, \beta = 0.3$ was therefore retained for the rest of the study, as it provides a more balanced contribution from the structural centrality term without a meaningful cost in fitness.

Table 7. Effect of α and β on fitness and violations ($n = 200$, dependency rate = 30%, 10 seeds).

α	β	Fitness	Violations
0.1	0.9	197.11	0
0.3	0.7	198.53	0
0.5	0.5	198.70	0
0.7	0.3	198.76	0
0.9	0.1	198.80	0
1.0	0.0	198.95	23.1

This result confirms the necessity of the structural component in the PPCO score. When $\beta = 0$, the score no longer contains descendant-based centrality and depends only on propagated activation probabilities. If some dependent rules are inactive or receive zero activation probability, the probability component may assign equal or non-discriminative scores to a predecessor and its successor. In that case, the sorting procedure has no structural information to enforce the required precedence relation, and violations may occur. By contrast, when $\beta > 0$, the descendant-based centrality term guarantees that each predecessor receives a strictly higher score than its dependent successors. This explains why all configurations with a non-zero structural component produce zero violations. Therefore, $\alpha = 1$ is excluded from the admissible PPCO configurations, despite its slightly higher raw fitness.

5.3.2. Effect of Dependency Rate

Figure 2 reports the average fitness $F(\sigma)$ of the four methods as the dependency rate varies for $n = 200$ rules. When the dependency rate is close to zero, all methods behave similarly because the ordering problem is weakly constrained. As the dependency rate increases, the gap between PPCO and the competing methods becomes much clearer. PPCO keeps a fitness close to the upper bound of the scale, while Greedy, SGM, and ZHTO-D1 degrade more rapidly. Beyond 50%, the number of edges stabilizes due to the out-degree constraint, and the relative ranking of the methods remains unchanged. This behavior shows that the propagated score remains effective even when the dependency structure becomes dense.

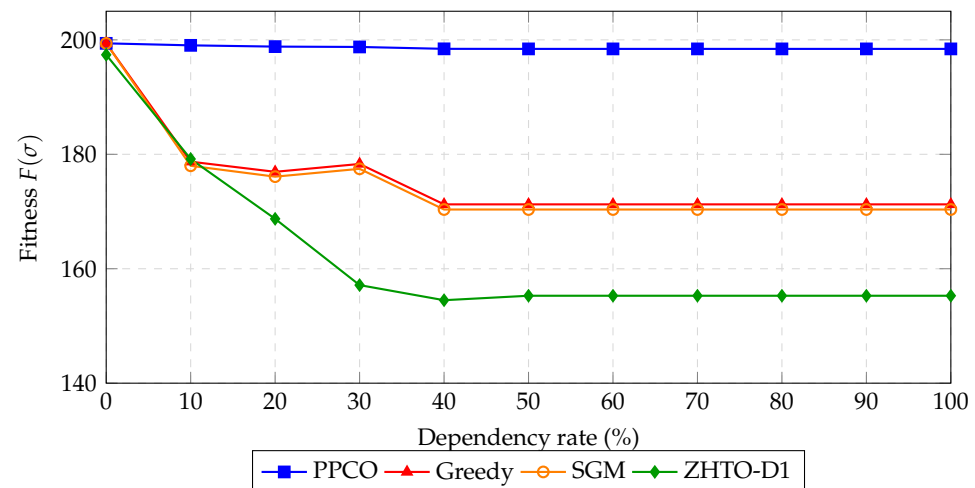


Figure 2. Fitness $F(\sigma)$ as a function of the dependency rate ($n = 200$, 10 seeds).

5.3.3. Effect of Rule Set Size

Figure 3 reports the average fitness for increasing rule set sizes at a fixed dependency rate of 30%. PPCO achieves a competitive fitness at every scale. The gain over Greedy is modest at $n = 100$, but becomes substantial for larger rule sets and remains above 13% from $n = 500$ onward. This shows that the benefit of combining propagated probability and structural centrality is not limited to small instances, but persists as the policy grows.



Figure 3. Fitness as a function of the synthetic rule set size.

At this scale, the Zipf distribution with parameter $a = 1.2$ concentrates 56.5% of the total activation probability in the top five rules alone. These highly dominant rules are ranked identically by both PPCO and Greedy, since their probability advantage is large enough to make the propagated score and the local frequency criterion agree on their placement. At the same time, the bottom 20 rules at $n = 100$ carry only 2.5% of the total probability combined, so their relative ordering has a negligible impact on $F(\sigma)$. The gain of PPCO over Greedy therefore depends primarily on the intermediate probability region, where the propagated score can reorder rules differently from a local criterion. At $n = 100$, this intermediate zone is too narrow to produce a large fitness difference. As n grows beyond 200, the probability mass redistributes more uniformly across a longer tail, the intermediate zone widens, and the structural advantage of PPCO becomes more visible. This explains why the gain increases again from $n = 200$ onward and stabilizes above 13% for larger rule sets.

5.3.4. Score-Coherence Evaluation

Figure 4 reports the average SCF values across policy sizes. PPCO achieves the lowest SCF in every configuration. The reduction compared with Greedy becomes particularly large for medium and large rule sets, reaching more than 95% at $n = 500$ and above. This indicates that the final ordering produced by PPCO remains strongly aligned with the dependency-aware score even as the size of the policy increases.

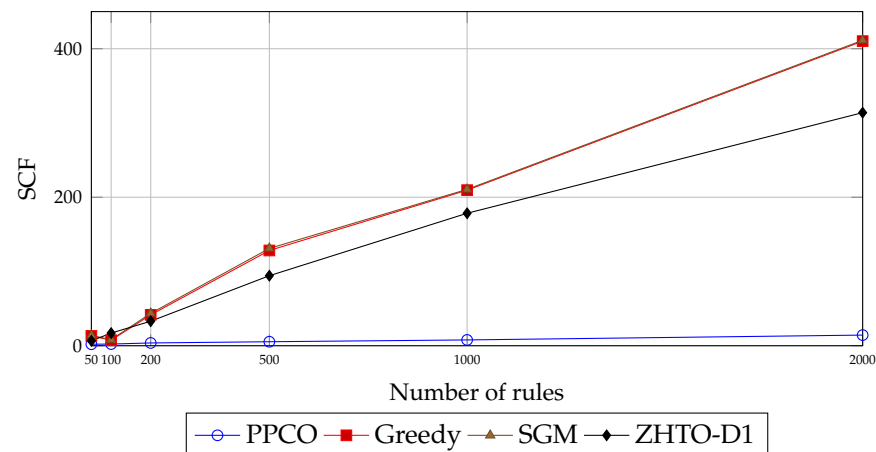


Figure 4. Score-Coherence Fitness as a function of the synthetic rule set size.

5.3.5. Execution Time

Table 8 reports the average execution time in seconds for synthetic rule sets of increasing size. PPCO remains competitive at small scales and becomes the fastest method as the number of rules increases. At $n = 50$, Greedy is slightly faster, which is expected since its local selection criterion involves very limited overhead on very small instances. From $n = 100$ onward, PPCO becomes consistently faster than all competing methods. The speedup over Greedy grows from $1.2\times$ at $n = 100$ to $15.9\times$ at $n = 2000$. This shows that PPCO combines strong ordering quality with low computational cost.

Table 8. Average execution time in seconds for synthetic rule sets.

n	PPCO	Greedy	SGM	ZHTO-D1
50	0.0006	0.0004	0.0008	0.0008
100	0.0009	0.0011	0.0020	0.0022
200	0.0015	0.0032	0.0046	0.0053
500	0.0025	0.0108	0.0139	0.0173
1000	0.0059	0.0494	0.0590	0.0732
2000	0.0116	0.1842	0.2015	0.2800

5.3.6. Extended Validation on ClassBench-ng Rule Sets

To address concerns about the scope of the benchmark validation, the evaluation was extended beyond the original fw1 profile. Thirteen ClassBench-ng instances were generated across four distinct seed profiles (fw1, fw2, ac11, ac12), covering rule set sizes from 50 to 892 rules and dependency densities ranging from 0 to more than 9 edges per rule. Table 9 reports the fitness obtained by all four methods on each instance, and Table 10 reports the corresponding Score-Coherence Fitness. Zero dependency violations were observed across all 13 instances and all four methods, confirming that policy correctness is preserved regardless of rule set size, dependency density, or seed profile. PPCO achieves the highest, or tied-highest, fitness on 4 of the 13 instances. On the remaining 9 instances, PPCO is outperformed in raw fitness by ZHTO-D1 or SGM, but the mar-

gin never exceeds 0.06%, which is not a practically meaningful difference given that all methods are evaluated under the same activation probability distribution. In contrast, PPCO achieves the lowest SCF on all 13 instances without exception, indicating that the orderings it produces remain the most internally consistent with the dependency-aware score across the entire benchmark suite, independently of the seed profile or rule set size.

Table 9. Extended ClassBench-ng validation across four seed profiles: Fitness $F(\sigma)$ (higher is better). Zero violations were observed for all methods on all instances.

Dataset	n	Edges	PPCO	Greedy	SGM	ZHTO-D1
fw1_50	50	0	50.00	50.00	50.00	50.00
fw1_100	88	67	84.86	7.96	7.96	80.84
fw1_200	122	137	121.79	119.78	119.78	28.87
fw1_500	240	837	235.19	139.58	139.95	235.22
fw1_1000a	398	1310	396.70	396.86	396.89	396.83
fw1_1000b	446	1400	443.78	394.90	394.46	443.86
fw1_2000	595	2699	592.38	546.57	545.48	592.70
fw2_1000	172	456	169.54	154.14	154.14	169.57
fw2_2000	295	844	293.15	289.02	285.42	293.19
acl1_1000	506	2194	502.43	427.25	427.24	502.28
acl1_2000	892	8533	881.18	621.44	679.72	881.21
acl2_1000	492	1814	490.65	490.83	490.83	490.84
acl2_2000	765	4374	763.48	761.66	761.04	763.53

Table 10. Extended ClassBench-ng validation across four seed profiles: Score-Coherence Fitness SCF (lower is better).

Dataset	n	PPCO	Greedy	SGM	ZHTO-D1
fw1_50	50	0.70	0.70	0.70	0.70
fw1_100	88	8.06	167.51	167.51	12.52
fw1_200	122	3.46	21.80	21.80	72.59
fw1_500	240	19.55	218.98	214.52	37.02
fw1_1000a	398	12.64	72.36	75.70	80.08
fw1_1000b	446	15.75	138.61	142.88	67.99
fw1_2000	595	26.84	198.71	203.40	122.33
fw2_1000	172	14.30	56.22	56.22	27.73
fw2_2000	295	16.46	61.86	82.34	76.95
acl1_1000	506	28.32	328.61	332.01	131.97
acl1_2000	892	96.87	1656.15	1462.28	460.30
acl2_1000	492	26.53	139.91	140.57	162.18
acl2_2000	765	63.61	276.18	266.00	244.34

5.3.7. Comparison with the Exact Optimal Ordering

The metrics used so far compare PPCO against deterministic baselines, but they do not indicate how close any of these methods is to the best possible ordering. To address this, the proposed score is additionally evaluated against the exact optimal solution on small instances, for which the true optimum can be computed without approximation. For a rule set of n rules, the optimal ordering σ^* is the one that maximizes the fitness $F(\sigma)$ among all orderings that respect the dependency constraints of the DAG. Finding σ^* by exhaustively testing every valid ordering is only feasible for small n , since the number of candidate orderings can grow as fast as $n!$. To make this tractable, σ^* is computed using a dynamic programming procedure over subsets of rules already placed, encoded as a bitmask. Each state stores the best partial fitness achievable for a given subset of placed rules, and the optimal ordering is reconstructed once all n rules have been placed. This approach has

a complexity of $O(2^n \cdot n)$, which is exponential but remains computable in a few seconds for n up to about 20, independently of how many dependencies are present. This procedure was applied to four sets of $n = 15$ rules, each extracted from a different real ClassBench-ng instance (fw1_500, fw1_1000, fw2_1000, ac11_1000). For each instance, a sub-DAG of 15 rules was selected so as to include the longest dependency chain present in the original file, ensuring that the extracted instance reflects a genuine multi-level dependency structure rather than a trivial one. Activation probabilities were assigned to the 15 rules following the same Zipf distribution used throughout the paper. Table 11 reports the fitness obtained by PPCO and the three baselines, expressed as a percentage of the exact optimal fitness $F(\sigma^*)$.

Table 11. Fitness as a percentage of the exact optimal fitness $F(\sigma^*)$, for four instances of $n = 15$ rules extracted from real ClassBench-ng files. 100% means the method reaches the true optimum. Zero violations were observed for all methods on all instances.

Source Instance	PPCO	Greedy	SGM	ZHTO-D1
fw1_500	100.00%	100.00%	100.00%	100.00%
fw1_1000	99.55%	33.07%	33.07%	99.55%
fw2_1000	100.00%	100.00%	100.00%	100.00%
ac11_1000	99.55%	41.44%	49.81%	99.55%

On two of the four instances (fw1_500 and fw2_1000), all four methods reach the exact optimum. Inspection of these two sub-DAGs shows that their dependency structure is shallow, with most rules directly dependent on a single dominant predecessor; in this simple setting, ranking by local activation probability alone is already sufficient to reach the optimal order, so no method has an opportunity to distinguish itself. On the other two instances (fw1_1000 and ac11_1000), which contain a genuine multi-level dependency chain, a clear separation appears. PPCO and ZHTO-D1 both reach 99.55% of the optimal fitness, while Greedy and SGM fall to 33–50% of the optimum. This confirms, on a small enough scale to allow exact verification, that the local selection criteria used by Greedy and SGM can lead to a substantial loss of ordering quality when dependency chains are non-trivial, while the propagated, non-local criteria used by PPCO and ZHTO-D1 remain close to optimal. PPCO and ZHTO-D1 are not distinguished from one another on these four instances, which is consistent with the comparable behavior already observed between the two methods on several of the extended ClassBench-ng instances in Section 5.3.

Table 12 reports the execution time of all four methods on the extended ClassBench-ng instances, averaged over 10 repeated measurements per instance. PPCO is faster than ZHTO-D1 on 12 of the 13 instances, with the only exception occurring on fw1_50, the single instance with no detected dependency, where all four methods produce identical orderings and timing differences reflect implementation-level overhead rather than algorithmic behavior. As shown in Figure 5, the relative speed advantage of PPCO over the baselines tends to grow with rule set size, although this trend is not strictly monotonic and fluctuates across individual instances depending on their specific dependency density.

Table 12. Execution time in seconds on the extended ClassBench-ng rule sets (mean of 10 repeats).

Dataset	n	PPCO	Greedy	SGM	ZHTO-D1
fw1_50	50	0.00025	0.00014	0.00034	0.00004
fw1_100	88	0.00037	0.00024	0.00062	0.00061
fw1_200	122	0.00057	0.00033	0.00096	0.00089
fw1_500	240	0.00142	0.00122	0.00289	0.00233

Table 12. Cont.

Dataset	<i>n</i>	PPCO	Greedy	SGM	ZHTO-D1
fw1_1000a	398	0.00242	0.00280	0.00504	0.00314
fw1_1000b	446	0.00246	0.00397	0.00639	0.00467
fw1_2000	595	0.00393	0.00650	0.01056	0.00702
fw2_1000	172	0.00089	0.00082	0.00172	0.00113
fw2_2000	295	0.00172	0.00186	0.00324	0.00256
acl1_1000	506	0.00336	0.00358	0.00742	0.00659
acl1_2000	892	0.01035	0.00844	0.01820	0.01096
acl2_1000	492	0.00303	0.00389	0.00682	0.00465
acl2_2000	765	0.00548	0.00850	0.01460	0.00965

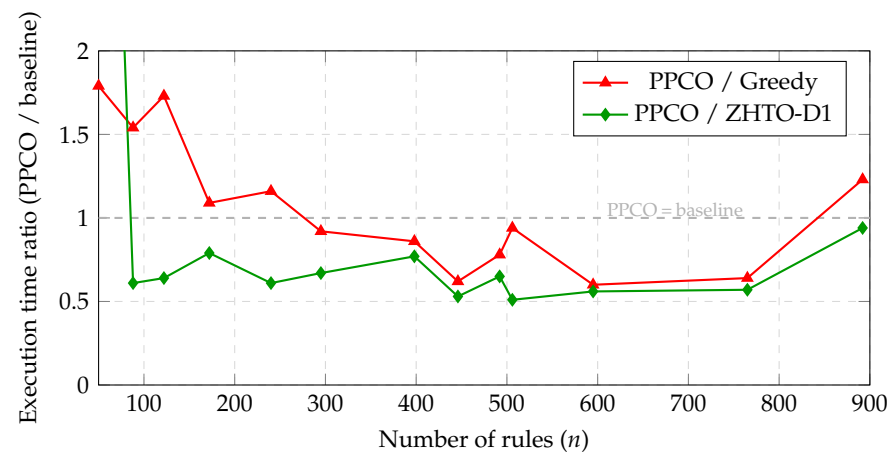


Figure 5. Ratio of PPCO execution time to Greedy and ZHTO-D1 execution time on the extended ClassBench-ng instances, as a function of rule set size. Values below 1 indicate that PPCO is faster. The outlier at $n = 50$ for ZHTO-D1 (ratio 6.25) corresponds to the only instance with zero detected dependencies, where timing differences reflect implementation overhead rather than algorithmic behavior.

5.4. Discussion

The results tell a fairly consistent story across all the experiments, under the experimental settings considered here and against the deterministic baselines implemented in this work. PPCO produces zero dependency violations in every valid configuration tested, on both the synthetic policies and the ClassBench-ng instances, and it keeps a strong ordering fitness across the range of policy sizes and dependency rates explored. This matters because it shows that the proposed score is not just a good ranking signal; it also respects the structural constraints of the policy. In other words, PPCO improves the quality of the ordering without weakening the dependency relations that keep the security behavior of the policy intact. The advantage of PPCO over the baselines grows as the dependency structure becomes denser or as the policy gets larger. This is expected given how the method was designed. When dependencies are sparse, local criteria still work reasonably well, because there is enough freedom in how rules can be ordered. However once dependencies pile up, local frequency alone stops being a reliable signal, and the position of a rule has to be understood in relation to the rules that depend on it. The propagated score does exactly this by combining activation probability with descendant-based centrality, and this is why PPCO keeps a high fitness even in cases where the other methods clearly degrade. The structural term in the score is not a minor addition. It becomes essential precisely when probabilities are equal, close together, or simply uninformative. The sensitivity analysis on α and β makes this point clear. When $\beta = 0$, the score collapses to propagated probability alone, and this becomes risky as soon as the traffic distribution is

sparse, since inactive rules can end up with zero probability. If two such rules are linked by a dependency, the probability term has nothing left to distinguish them, and the resulting order may violate that dependency. Keeping $\beta > 0$ removes this risk entirely, because the centrality term always gives a strictly higher structural value to a predecessor than to its descendants. Thus, $\beta > 0$ is not just about refining the ranking, it is what keeps the ordering safe when the available traffic information is incomplete or weakly discriminative. The configuration $\alpha = 0.7$, $\beta = 0.3$ used throughout the paper reflects this balance between following the traffic signal and keeping enough structural weight to avoid these failure cases. Score-Coherence Fitness adds another angle to this picture. PPCO reaches the lowest SCF on every single instance tested, both synthetic and ClassBench-ng, and the gap with the other methods tends to widen as the policy grows. This tells us something specific: the orderings produced by PPCO stay closely aligned with the score that generated them. PPCO is not only doing well on the fitness metric, it is also internally consistent with its own ranking logic, which supports the idea that this is a structured ranking method rather than a heuristic that happens to work. The runtime results reinforce this picture from a practical angle. PPCO stays lightweight and becomes clearly faster than the baselines as the policy size increases, which matters for real deployments where a rule ordering method needs to remain usable on large policies without turning into an expensive optimization step on its own. The extended validation on ClassBench-ng tells a more nuanced story than the synthetic experiments, and it is worth being precise about it. PPCO does not get the highest raw fitness on every single instance. It achieves the best, or tied-best, fitness on 4 of the 13 instances, and on the remaining 9 it stays within a very small margin of whichever baseline performs best there, never more than 0.06% behind. What does stay consistent across all 13 instances is the SCF: PPCO achieves the lowest value every time, which means its orderings remain the most coherent with the dependency-aware score, regardless of the seed profile or the size of the rule set. Zero violations hold throughout as well. So the overall picture from ClassBench-ng is not that PPCO dominates every metric on every instance, but that it stays close to the best in fitness while being clearly the most internally consistent method, and it does this without ever compromising on correctness. A point worth discussing in more detail is the comparison between PPCO and ZHTO-D1, since the two methods often land close to each other in fitness, and both reach near-optimal results on the small instances tested against the exact optimum. Despite this similarity in ordering quality, PPCO is faster than ZHTO-D1 on 12 of the 13 extended ClassBench-ng instances, and the gap tends to widen as the rule sets get larger. This suggests that the advantage of PPCO is not limited to how good the final ordering is, it also shows up in how the ordering is built. PPCO reaches comparable, and often better, ordering quality through a single global sort, while ZHTO-D1 relies on a threshold-based split of rules followed by a layer assignment step, which adds computational overhead without a corresponding gain in quality. The most striking individual result is on the fw1_100 instance, where PPCO achieves a gain of +965% over Greedy. This instance has 88 rules and 67 dependency edges, which works out to a dependency density of roughly 76%, so the structure is heavily constrained. Under the Zipf-distributed probabilities used here, a small number of rules carry most of the weight, while several predecessor rules in the dependency chain have low local probability. Greedy picks, at each step, whichever admissible rule has the highest local probability, so it never breaks a dependency, but it has no way of knowing that a low-probability predecessor is gating a chain of much more active descendants. The result is that these predecessors get pushed late in the ordering, dragging the whole chain down with them. PPCO does not have this blind spot, since its propagated probability term sums up the activation probability of every descendant of a rule, so a predecessor with a low local probability but many active descendants still ends

up with a high score and gets placed early. The size of the gain on fw1_100 is therefore not some artifact of how the benchmark was generated; it follows directly from the mismatch between a local view of importance and a dependency-aware one, and this mismatch is at its sharpest exactly when the dependency density is this high. Taken together, these results support the central claim of the paper. Combining propagated probability with descendant-based centrality gives a deterministic ordering method that stays correct by construction, performs competitively or better than the strongest baselines, and remains internally consistent with its own scoring logic, all while keeping the computational cost low enough to scale to large policies.

6. Conclusions

This paper introduced PPCO, a deterministic method for firewall rule ordering based on propagated probability and descendant-based centrality. The main idea of the method is to evaluate each rule not only through its local activation probability but also through its structural influence inside the dependency graph. In this way, PPCO captures two essential dimensions of rule importance: traffic relevance and dependency-aware influence. This gives the method a clear originality compared with local frequency-based rankings and iterative search-based approaches. The proposed formulation leads to a simple and interpretable ordering mechanism. Each rule receives a global score that reflects both its own activity and the contribution of the descendant rules that depend on it. The final policy is then obtained through a direct descending sort of these scores. This design avoids stochastic exploration, iterative refinement, and dependence on initialization quality. As a result, PPCO remains deterministic, stable, and easy to reproduce. Its robustness also comes from the fact that the score is consistent with the dependency structure, which allows the method to preserve valid rule precedence in all admissible configurations. The experimental results confirm the effectiveness of this design. Across synthetic policies ranging from 50 to 2000 rules, PPCO consistently achieved competitive fitness values among the evaluated deterministic baselines while maintaining zero dependency violations. The method remained stable as the dependency rate increased and continued to outperform the compared methods under the tested structural constraints. The parameter analysis further showed that the combination $\alpha = 0.7$ and $\beta = 0.3$ provides the best valid balance between probabilistic relevance and structural centrality. In addition, PPCO obtained the lowest score-coherence values in all tested settings, which confirms that the generated orders remain strongly aligned with the proposed importance model. The runtime evaluation also showed that PPCO scales efficiently and becomes significantly faster than the compared methods as the rule set size grows. The extended validation on ClassBench-ng rule sets, covering four seed profiles and rule set sizes up to 892 rules, confirmed that PPCO remains within a small margin of the strongest baseline in raw fitness and achieves the lowest score-coherence values on every tested instance, while preserving zero dependency violations throughout. PPCO also proved faster than ZHTO-D1, the closest competitor in ordering quality, on 12 of the 13 extended benchmark instances, with the gap tending to grow with rule set size. Finally, the comparison against exact optimal orderings on small instances showed that PPCO reaches 99.55% or more of the true optimum on every tested instance, while Greedy and SGM fall to 33–50% of the optimum when the dependency structure is non-trivial. Taken together, these results show that PPCO provides a robust and practical solution for dependency-aware firewall rule ordering. Its main strength lies in the fact that it remains simple, deterministic, and structurally grounded, while still capturing a non-local notion of rule importance. The method therefore offers a strong alternative to iterative optimization techniques when correctness, stability, and low execution cost are required at the same time. Future work also considers extending the dependency-aware

scoring framework to more dynamic network environments, including cloud-based firewall architectures and zero-trust security models, where rule ordering constraints and traffic distributions may evolve continuously.

Author Contributions: Conceptualization, F.B. and D.L.; methodology, F.B. and D.L.; software, F.B.; validation, F.B. and D.L.; formal analysis, F.B.; investigation, F.B.; writing—original draft preparation, F.B.; writing—review and editing, F.B. and D.L.; supervision, D.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The code supporting the results reported in this study, including the PPCO implementation, the reimplemented baselines (Greedy, SGM, ZHTO-D1), the synthetic rule set generator, and the random seeds used in all experiments, is publicly available at: <https://github.com/fadwabezzazi/ppco-firewall-ordering> (accessed on 22 June 2026).

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

References

1. Alsmadi, I. Misconfiguration in Firewalls and Network Access Controls: Literature Review. *Future Internet* **2021**, *13*, 283. [CrossRef]
2. Fulp, E.W. Optimization of Network Firewall Policies Using Directed Acyclical Graphs. In *Proceedings of the IEEE Internet Management Conference*; Citeseer: Princeton, NJ, USA, 2005.
3. Dobrescu, M.; Argyraki, K. Dynamic Rule-Ordering Optimization for High-Speed Firewall Filtering. In *Proceedings of the ACM Symposium on Information, Computer and Communications Security*; Association for Computing Machinery: New York, NY, USA, 2006.
4. El-Alfy, E.S.M. A Heuristic Approach for Firewall Policy Optimization. In *Proceedings of the ICACT 2007*; IEEE: Piscataway, NJ, USA, 2007; pp. 851–856.
5. Ben Neji, N.; Bouhoula, A. Towards Safe and Optimal Filtering Rule Reordering for Complex Packet Filters. In *Proceedings of the 2011 5th International Conference on Network and System Security (NSS)*, Piscataway, NJ, USA, 6–8 September 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 153–160. [CrossRef]
6. Horn, W.A. Single-Machine Job Sequencing with Treelike Precedence Ordering and Linear Delay Penalties. *SIAM J. Appl. Math.* **1972**, *23*, 189–202. [CrossRef]
7. Kirkpatrick, S.; Gelatt, C.D.; Vecchi, M.P. Optimization by Simulated Annealing. *Science* **1983**, *220*, 671–680. [CrossRef] [PubMed]
8. Holland, J.H. *Adaptation in Natural and Artificial Systems*, 2nd ed.; MIT Press: Cambridge, MA, USA, 1992.
9. Zhigljavsky, A. Random Search for Global Optimization. In *Encyclopedia of Optimization*; Pardalos, P.M., Prokopyev, O.A., Eds.; Springer: Cham, Switzerland, 2023. [CrossRef]
10. Bezzazi, F.; Lotfi, D. Optimizing Firewall Policy Rule Ordering Using Discrete Cuckoo Search Algorithm. In *Proceedings of the WINCOM 2025*; IEEE: Piscataway, NJ, USA, 2025; pp. 1–6.
11. Al-Shaer, E.S.; Hamed, H.H. Discovery of Policy Anomalies in Distributed Firewalls. In *Proceedings of the IEEE INFOCOM 2004*, Piscataway, NJ, USA, 7–11 March 2004; IEEE: Piscataway, NJ, USA, 2004; Volume 4, pp. 2605–2616.
12. Abedin, M.A.U.; Nessa, S.; Khan, L.; Thuraisingham, B. Detection and Resolution of Anomalies in Firewall Policy Rules. In *Data and Applications Security XX*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2006; Volume 4127, pp. 15–29. [CrossRef]
13. Hu, H.; Ahn, G.J.; Kulkarni, K. Detecting and Resolving Firewall Policy Anomalies. *IEEE Trans. Dependable Secur. Comput.* **2012**, *9*, 318–331. [CrossRef]
14. Lu, N.; Yang, Y. Application of evolutionary algorithm in performance optimization of embedded network firewall. *Microprocess. Microsyst.* **2020**, *76*, 103087. [CrossRef]
15. Coscia, A.; Dentamaro, V.; Galantucci, S.; Maci, A.; Pirlo, G. An Innovative Two-Stage Algorithm to Optimize Firewall Rule Ordering. *Comput. Secur.* **2023**, *134*, 103423. [CrossRef]
16. Coscia, A.; Maci, A.; Tamma, N. FROG: A Firewall Rule Order Generator for faster packet filtering. *Comput. Netw.* **2025**, *257*, 110962. [CrossRef]
17. Yang, X.S.; Deb, S. Cuckoo Search via Lévy Flights. In *Proceedings of the World Congress on Nature and Biologically Inspired Computing (NaBIC 2009)*, Piscataway, NJ, USA, 9–11 December 2009; IEEE: Piscataway, NJ, USA, 2009; pp. 210–214.
18. Ouaraab, A. *Discrete Cuckoo Search for Combinatorial Optimization*; Springer: Singapore, 2020. [CrossRef]

19. Yang, X.S.; He, X. Cuckoo Search and Firefly Algorithm: Hybridization and Applications. In *Proceedings of the 2013 IEEE Congress on Evolutionary Computation (CEC)*; IEEE: Piscataway, NJ, USA, 2013; pp. 82–89.
20. Semenov, S.; Zhang, L.; Cao, W.; Bulba, S.; Babenko, V.; Davydov, V. Development of a Fuzzy GERT Model for Investigating Common Software Vulnerabilities. *East.-Eur. J. Enterp. Technol.* **2021**, *6*, 6–18. [[CrossRef](#)]
21. Bang-Jensen, J.; Gutin, G. *Digraphs: Theory, Algorithms and Applications*, 2nd ed.; Springer: London, UK, 2008.
22. Matoúšek, J.; Antichi, G.; Lučanský, A.; Moore, A.W.; Kořenek, J. ClassBench-ng: Recasting ClassBench After a Decade of Network Evolution. In *Proceedings of the 2017 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, Piscataway, NJ, USA, 18–19 May 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 204–216.
23. Breslau, L.; Cao, P.; Fan, L.; Phillips, G.; Shenker, S. Web Caching and Zipf-like Distributions: Evidence and Implications. In *Proceedings of the IEEE INFOCOM 1999, Piscataway, NJ, USA, 21–25 March 1999*; IEEE: Piscataway, NJ, USA, 1999; Volume 1, pp. 126–134.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.