2-2019

FPC: A New Approach to Firewall Policies Compression

Yuzhu Cheng
Central South University

Weiping Wang
Central South University, wpwang@mail.csu.edu.cn

Jianxin Wang
Central South University

Haodong Wang
Cleveland State University, H.WANG96@csuohio.edu

Publisher's Statement

FPC: A New Approach to Firewall Policies Compression

Yuzhu Cheng, Weiping Wang, Jianxin Wang, and Haodong Wang

Abstract: Firewalls are crucial elements that enhance network security by examining the field values of every packet and deciding whether to accept or discard a packet according to the firewall policies. With the development of networks, the number of rules in firewalls has rapidly increased, consequently degrading network performance. In addition, because most real-life firewalls have been plagued with policy conflicts, malicious traffics can be allowed or legitimate traffics can be blocked. Moreover, because of the complexity of the firewall policies, it is very important to reduce the number of rules in a firewall while keeping the rule semantics unchanged and the target firewall rules conflict-free. In this study, we make three major contributions. First, we present a new approach in which a geometric model, multidimensional rectilinear polygon, is constructed for the firewall rules compression problem. Second, we propose a new scheme, Firewall Policies Compression (FPC), to compress the multidimensional firewall rules based on this geometric model. Third, we conducted extensive experiments to evaluate the performance of the proposed method. The experimental results demonstrate that the FPC method outperforms the existing approaches, in terms of compression ratio and efficiency while maintaining conflict-free firewall rules.

Key words: firewall; firewall policy; network security; firewall rules compression

1 Introduction

Firewalls are critical components of network security and are deployed at all the entrances between a private network and the Internet to monitor all incoming and outgoing packets. The function of a firewall is to examine the field values of every packet and decide whether to accept or discard a packet according to the firewall policies. The policy is specified as a sequence of rules, each of which has a predicate over some packet header fields and a decision to be performed upon the packets that match the predicate. A packet can be viewed as a tuple with a finite number of the fields such as the source Internet Protocol (IP) address, destination IP address, source port number, destination port number, and protocol type. When a packet arrives at a firewall, the firewall searches for the first (i.e., the highest priority) rule with which the packet matches, and executes the decision of the rule. In general, the decision of a rule is *accept* or *discard*. Two firewalls are semantically equivalent to each other if and only if they have the same decision for every possible packet. In this paper, we consider the enhanced Firewall Policies Compression (FPC) problem: Given a firewall f, generate another firewall f' that is semantically equivalent to f but has fewer conflict-free rules.

In high-speed firewalls, filtering is performed by special hardware (i.e., Ternary Content Addressable Memory (TCAM)[1]), which is not only expensive but also constraints on the size of firewall rule sets. Thus, a natural optimization criterion for firewall

rule sets is to minimize their sizes while maintaining the semantic equivalence. This will also reduce the maximum delay in the sequential rules evaluation by the firewalls. Another optimization criterion is to ensure the rules are conflict-free based on the following two points[2]: First, because of the conflicts among rules and the resulting order sensitivity, firewall rules are logically entangled, and the correct ordering is critical but difficult. The semantics of any rule in a firewall cannot be correctly understood without examining previously listed rules. Second, if a firewall administrator unintentionally swaps any two conflicting rules, wrong actions may be performed, resulting in irreparable or tragic consequences.

In Ref. [2], Cheng et al. presented an approach for designing firewalls based on a multidimensional matrix. In their proposed method, a Firewall Design Matrix (FDM) is first designed. Then, a construction algorithm and a generation algorithm are applied in the FDM to generate and compress the target firewall rules, which are disjoint and non-conflicting. However, we find that the rules can be further compressed when rules with the same decision are allowed to overlap; here, the overlapping does not affect the semantics of the rules. Figure 1 illustrates an example of this instance. Assuming there are three firewall rules r_1: $F_1 \in [1,2] \wedge F_2 \in [4,7] \rightarrow accept$, r_2: $F_1 \in [3,4] \wedge F_2 \in [2,8] \rightarrow accept$, and r_3: $F_1 \in [5,7] \wedge F_2 \in [4,7] \rightarrow accept$, as shown in Fig. 1a, since these rules have the same decision, and are allowed to overlap, they can be further compressed to two rules: r_1: $F_1 \in [1,7] \wedge F_2 \in [4,7] \rightarrow accept$, and r_2: $F_1 \in [3,4] \wedge F_2 \in [2,8] \rightarrow accept$, as shown in Fig. 1b.

9										
8				$1, r_2$	$1, r_2$					
7		$1, r_1$	$1, r_1$	$1, r_2$	$1, r_2$	$1, r_3$	$1, r_3$	$1, r_3$		
6		$1, r_1$	$1, r_1$	$1, r_2$	$1, r_2$	$1, r_3$	$1, r_3$	$1, r_3$		
5		$1, r_1$	$1, r_1$	$1, r_2$	$1, r_2$	$1, r_3$	$1, r_3$	$1, r_3$		
4		$1, r_1$	$1, r_1$	$1, r_2$	$1, r_2$	$1, r_3$	$1, r_3$	$1, r_3$		
3				$1, r_2$	$1, r_2$					
2				$1, r_2$	$1, r_2$					
1										
0										
F_2/F_1	0	1	2	3	4	5	6	7	8	9

(a) Three rules before compression.

9										
8				$1, r_2$	$1, r_2$					
7		$1, r_1$	$1, r_1$	$1, r_1/r_2$	$1, r_1/r_2$	$1, r_1$	$1, r_1$	$1, r_1$		
6		$1, r_1$	$1, r_1$	$1, r_1/r_2$	$1, r_1/r_2$	$1, r_1$	$1, r_1$	$1, r_1$		
5		$1, r_1$	$1, r_1$	$1, r_1/r_2$	$1, r_1/r_2$	$1, r_1$	$1, r_1$	$1, r_1$		
4		$1, r_1$	$1, r_1$	$1, r_1/r_2$	$1, r_1/r_2$	$1, r_1$	$1, r_1$	$1, r_1$		
3				$1, r_2$	$1, r_2$					
2				$1, r_2$	$1, r_2$					
1										
0										
F_2/F_1	0	1	2	3	4	5	6	7	8	9

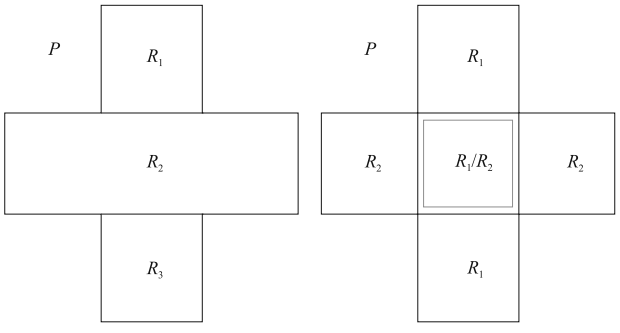
(b) Two rules after compression.

Fig. 1 A compressing example, r_1: $F_1 \in [1,2] \wedge F_2 \in [4,7] \rightarrow accept$, r_2: $F_1 \in [3,4] \wedge F_2 \in [2,8] \rightarrow accept$, and r_3: $F_1 \in [5,7] \wedge F_2 \in [4,7] \rightarrow accept$.

On the other hand, rectilinear polygons frequently arise in Very Large Scale Integration (VLSI) layout and artwork analyses as well as computer graphics[3]. Often, functions are more easily performed on a rectilinear polygon by considering the polygon as being composed of several rectangles. A set T of rectangles is said to cover the polygon P if P is the union of T. If the rectangles in T are disjoint, then we call the set T to be a *partition* of P.

A Minimal Overlapping Cover (MOC) of a rectilinear polygon P is defined as the *cover* of P that has the fewest number of rectangles. A Minimal Non-overlapping Cover (MNC) of a rectilinear polygon P is defined as a *partition* of P with the minimum number of rectangles[4]. Figures 2a and 2b show a rectilinear polygon P with $|\mathrm{MNC}(P)| = 3$ and $|\mathrm{MOC}(P)| = 2$, respectively.

As defined in Ref. [2], the FDM method can be seen as the MNC problem of a multidimensional rectilinear polygon, and the target firewall rules are disjoint and conflict-free. However, if overlapping is allowed, it can be regarded as an MOC problem in a multidimensional space. In general, $|\mathrm{MOC}(P)|$ is less than $|\mathrm{MNC}(P)|$[4], as shown in Fig. 2. This means if the firewall rules compression problem is transformed into an MOC problem, the number of rules can be further reduced with a higher degree compared with the FDM method. In this paper, we propose a new method to transform the firewall rules compression and a novel algorithm that solves



(a) (b)

Fig. 2 Rectangular covers of a rectilinear polygon. (a) Non-overlapping cover (size = 3) and (b) overlapping cover (size = 2).

the corresponding MOC problem. This is our main innovation compared to prior researches[2,5–9] on the firewall rules compression. However, to the best of our knowledge, current studies of the MOC problem are concentrated on a two-dimensional space[9–13]. Therefore, we need to consider how to deal with this problem in a multidimensional case, which has been proved to be NP-hard[5].

The main contributions of this study can be summarized as follows:

(1) We present a new approach in which a geometric model is first constructed to transform the firewall rules compression problem into a multidimensional MOC problem.

(2) We propose a heuristic algorithm, FPC, to solve the multidimensional MOC problem, which is totally different from prior methods.

(3) We evaluated the performance of our approach by conducting extensive experiments on synthetic firewalls with different sizes. The experimental results demonstrate that FPC outperforms the existing approaches, in terms of compression ratio and efficiency, while maintaining conflict-free firewall rules.

The rest of this paper is organized as follows. Related works are presented in Section 2. In Section 3, we define terms related to our approach and present the algorithm for compressing the original firewall rules. In Section 4, we give the compression results of synthetic firewall rules. Finally, conclusion is drawn in Section 5.

2 Related Work

In this study, we convert the FPC problem to an MOC problem in a multidimensional space. To our knowledge, except several special cases, this problem in the general setting has never been studied in prior works. The optimal polynomial time algorithms for a one-dimensional case have been developed in Ref. [14]. In addition, a two-dimensional rules compression problem has been proved to be NP-hard[6]. Applegate et al.[5] proposed an optimal polynomial time algorithm for a two-dimensional problem when there are only two decisions and all rules must be strip rules, and then with this result, they created an approximation algorithm for the general two-dimensional problem. However, it is not clear how to apply their ideas to a space with more than two dimensions.

Daly et al.[7] presented Diplomat to address the multidimensional access control rules compression problem, where, the key idea is to transform higher-dimensional target patterns to lower-dimensional patterns by dividing the original pattern into a series of hyper-planes. Two adjacent planes are then selected, and their differences are resolved by adding rules to specify where the two planes differ. After resolution, the two planes are compatible and can be merged into a single plane. In this approach, each difference between two adjacent planes requires specification by additional rules. Therefore, the compression ratio is largely affected by the differences among the rules.

Cheng et al.[2] presented the FDM approach for designing firewalls. In their proposed method, the original firewall rules are mapped into a set of unit spaces in a multidimensional matrix. These unit spaces are disjoint from each other, and this approach can be seen as an MNC problem[4] of a multidimensional space. Based on this research, we transform the task of firewall rules compression to a multidimensional MOC problem. Given a rectilinear polygon P, the MOC problem requires determining the minimum number of axis-parallel rectangles whose union covers P. For the MOC problem on a two-dimensional space, researchers have proposed several distinguished schemes[5,9–13]. However, only little has been done for cases with over two dimensions, and real-life firewalls are typically five-dimensional. In this paper, we propose a heuristic approach for the first time to cover the multidimensional rectilinear polygon with the minimum number of rectilinear hyper-rectangles.

3 Proposed Approach

In this section, we first define the relevant concepts of our approach, and then we present our FPC algorithm. Table 1 lists the notations used in this paper.

Table 1 Notations used in this paper.

Notation	Description
f	Firewall
FDM	Firewall design matrix model
MOC	Minimal overlapping cover
MNC	Minimal non-overlapping cover
F_i	i-th dimension
$D(F_i)$	Domain of F_i
M_k	A k-dimensional matrix
u,v	Unit space
$R(u,v,F_i)$	Spatial relation between u and v in F_i
c_i	Grid cell
REC	Rectangle
P	Polygon
L_k	Linked list

3.1 Preliminaries

A firewall rule generally has the form ⟨predicate⟩ → ⟨decision⟩, and any ⟨predicate⟩ of the firewall rules has the form "$F_1 \in D(F_1) \wedge \cdots \wedge F_k \in D(F_k)$". A field F_i is a variable whose value is taken from a predefined interval of non-negative integers, called the F_i domain and denoted by $D(F_i)$. For example, the domain of the source address in an IP packet is $[0, 2^{32}-1]$.

FDM. An FDM over fields $(F_1, F_2, \ldots, F_k)$ is a k-dimensional matrix M_k. Each dimensional coordinate in M_k is represented by F_i, and the corresponding value is within a range of $[0, D(F_i)]$, $1 \leqslant i \leqslant k$. Any of the firewall rules with the form ⟨predicate⟩ → ⟨decision⟩ can be formally represented as an M_k. In other words, the predicate of each rule can be expressed as a multidimensional rectangular region in M_k, and the region value reflects the rule's decision: value = 0 (if ⟨*decision*⟩ = *discard*), or value = 1 (if ⟨*decision*⟩ = *accept*).

Unit space. An M_k may have a set of multidimensional rectangular regions with a value of "1", each of which can be represented by its low vertex coordinates and the corresponding distance, denoted as $[(l_1, l_2, \ldots, l_k)(d_1, d_2, \ldots, d_k)]$, where l_i is the vertex coordinate representing the lower bound of $D(F_i)$, and d_i is the distance representing the size of $D(F_i)$. These regions are referred as the unit spaces of M_k.

The procedure of mapping a firewall rule r_i to a k-dimensional matrix M_k is presented in Ref. [4]. Figures 3a and 3b illustrate this process. Assuming there are four firewall rules r_1: $F_1 \in [0, 2] \wedge F_2 \in [0, 1] \rightarrow$*discard*, r_2: $F_1 \in [3, 5] \wedge F_2 \in [0, 1] \rightarrow$*accept*, r_3: $F_1 \in [5, 5] \wedge F_2 \in [3, 5] \rightarrow$*discard*, and r_4: $F_1 \in [0, 5] \wedge F_2 \in [1, 5] \rightarrow$*accept*, and we map rule r_i to M_k in reverse order, while assigning the value of mapping region as "0" or "1" based on if ⟨*decision*⟩ is *discard* or *accept*. Figure 3a shows the FDM to which rule r_4 is mapped. There is only one unit space in this figure, denoted as [(0, 1), (6, 5)]. Then after r_3, r_2 and r_1 are mapped, the resulting FDM is shown in Fig. 3b, and the final three unit spaces are [(0, 2)(5, 4)], [(3, 0)(3, 2)], and [(5, 2)(1, 1)], corresponding to u_1, u_2, and u_3, respectively.

For ease of representation, we define the spatial relation between two unit spaces in a certain dimension. Given any two unit spaces u and v, the *spatial relation* between them in a certain dimension F_i can be represented as $R(u,v,F_i)$, where the value is an element of *adjacent*, *disjunct*, *crossed*, *covered*, *included*, and *equivalent*, as shown in Fig. 4.

Let $u = (l_1^{(u)}, \cdots, l_k^{(u)})(d_1^{(u)}, \cdots, d_k^{(u)})$, $v = (l_1^{(v)}, \cdots, l_k^{(v)})(d_1^{(v)}, \cdots, d_k^{(v)})$,

(1) If $l_i^{(u)} + d_i^{(u)} < l_i^{(v)} \| l_i^{(u)} > l_i^{(v)} + d_i^{(v)}$, $R(u, v, F_i) =$ *disjunct*;

(2) If $l_i^{(u)} + d_i^{(u)} = l_i^{(v)} \parallel l_i^{(u)} = l_i^{(v)} + d_i^{(v)}$, $R(u, v, F_i) =$ *adjacent*;

(3) If $(l_i^{(u)} < l_i^{(v)} < l_i^{(u)} + d_i^{(u)} < l_i^{(v)} + d_i^{(v)}) \parallel (l_i^{(v)} < l_i^{(u)} < l_i^{(v)} + d_i^{(v)} < l_i^{(u)} + d_i^{(u)})$, $R(u, v, F_i) =$ *crossed*;

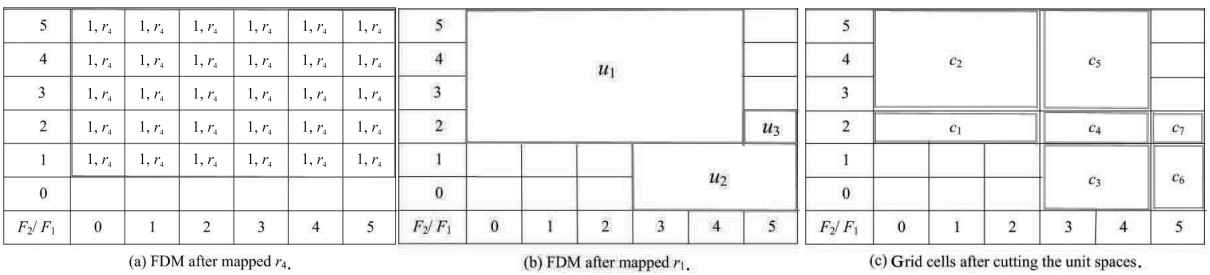
(4) If $l_i^{(v)} < l_i^{(u)} < l_i^{(u)} + d_i^{(u)} < l_i^{(v)} + d_i^{(v)}$, $R(u, v, F_i) =$ *included*;

(5) If $l_i^{(u)} < l_i^{(v)} < l_i^{(v)} + d_i^{(v)} < l_i^{(u)} + d_i^{(u)}$, $R(u, v, F_i) =$ *covered*;

(6) If $l_i^{(u)} = l_i^{(v)} \,\&\, d_i^{(u)} = d_i^{(v)}$, $R(u, v, F_i) =$ *equivalent*.

In general, given any two unit spaces u and v, if they are disjunct in a certain dimension or are adjacent in two or more dimensions, they are independent of each other. Accordingly, any unit space in an independent unit space set is independent of that in another independent unit space sets.

Grid cell. In a certain dimension F_i, if two unit spaces u and v satisfy $R(u, v, F_i) =$*included*, then we cut v into two or three sub-unit spaces, in which there exists a sub-unit space v' that satisfies $R(u, v', F_i) =$ *equivalent*. A cutting operation is iteratively conducted on all the unit spaces, until there are only disjunct or adjacent spatial relations between any two sub-unit spaces in all the k dimensions. We call these sub-unit



(a) FDM after mapped r_4. (b) FDM after mapped r_1. (c) Grid cells after cutting the unit spaces.

Fig. 3 An intuitive example of mapping and the corresponding grid cells, r_1–r_4.

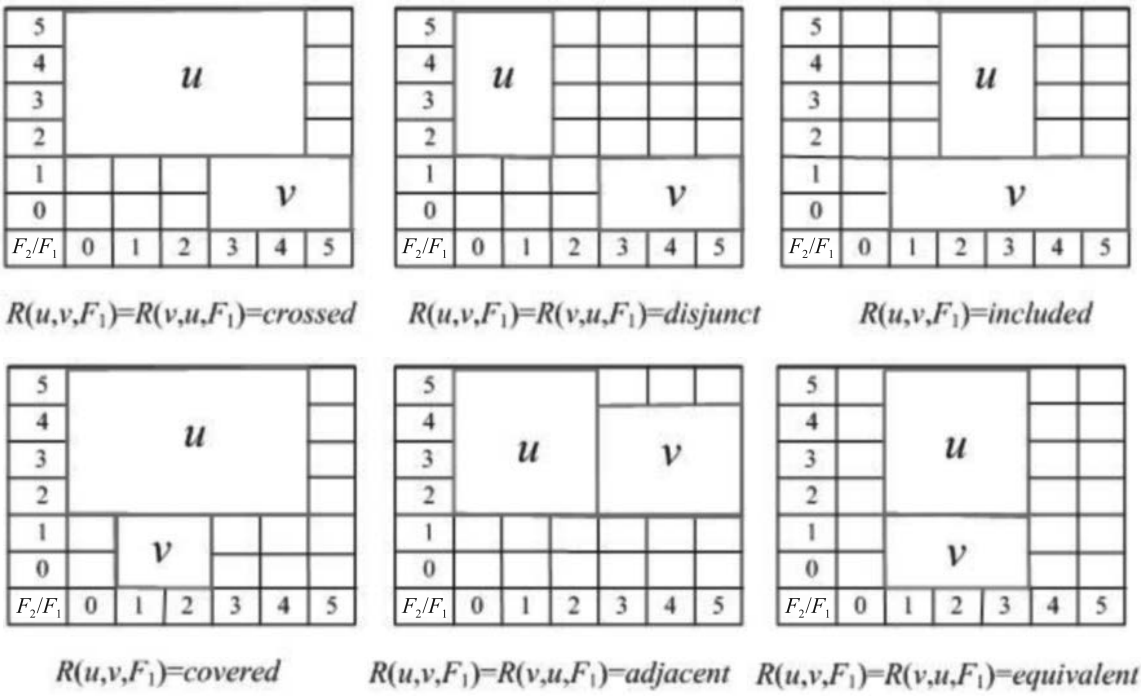


Fig. 4 Spational relation between two unit spaces u and v in dimension F_1.

spaces grid cells.

For any grid cell, the following two properties are satisfied:

Property 1. Given any two grid cells in a k-dimensional space, if they are adjacent in a certain dimension, then their coordinate values are identical in the rest $k-1$ dimensions.

Proof: Proof by contradiction. Given two grid cells c_1 and c_2 in a k-dimensional space, they are adjacent in the first dimension. Suppose their coordinate values are not completely identical in the rest $k-1$ dimensions; for example, the coordinate value of c_1 is less than that of c_2 in dimension F_i $(1 \leqslant i \leqslant k)$. According to the definition of grid cell, we will let the coordinate value of c_1 be the secant to partition c_2 in dimension F_i. Then, at this time, c_2 is equal to c_1 in dimension F_i after the partition. This contradicts the assumption that the coordinate value of c_1 is less than that of c_2 in dimension F_i. Therefore, given any two grid cells in k-dimensional space, if they are adjacent in a certain dimension, then their coordinate values are identical in the rest $k-1$ dimensions.

Property 2. Given any grid cell c_i in a k-dimensional space, if it has an adjacent grid cell c_j in a certain dimension, then c_i can be expanded to c_j to form a hyper-rectangle.

Proof: Given two grid cells c_i and c_j in a k-dimensional space, if they are adjacent in the dimension F_t, then their coordinate values are identical in the rest $k-1$ dimensions based on Property 1. Since the upper coordinate value of c_i is equal to the lower coordinate value of c_j in dimension F_t, these two grid cells can be combined in dimension F_t to form a hyper rectangle.

For example, there are seven grid cells from c_1 to c_7 in Fig. 3c, which are partitioned from the unit spaces u_1, u_2, and u_3 (Fig. 3b). The dimension of these grid cells is equivalent to that of the unit spaces, and each grid cell is either disjunct or adjacent to each other. Taking the grid cell c_1 as an example, $R(c_1, c_2, F_2) = adjacent$ and $R(c_1, c_2, F_1) = equivalent$; therefore, these two grid cells can be combined to form a rectangle c_1-c_2.

3.2 Methodologies

In this section, we formally describe our FPC method.

3.2.1 Problem description

First, a geometric model is first constructed to transform the FPC problem into an MOC problem in a multidimensional space.

Specifically, using the FDM method, we map the firewall rules into a multidimensional matrix to form a set of unit spaces, which constitute several multidimensional rectilinear polygons. In the multidimensional space, any firewall rule can be described by a multidimensional rectangle. Therefore,

the firewall rules compression problem can be transformed to use the minimum number of hyper-rectangles to cover these rectilinear polygons (MOC).

As mentioned in Section 1, the MOC problem is NP-hard in two or more dimensions[5], and to the best of our knowledge, there are no heuristic algorithms on MOC problems in a multidimensional case.

3.2.2 FPC heuristic algorithm

Considering that the solution of MOC problem in multidimensional space is NP-hard, we propose a heuristic method — FPC. The main idea of this method is as follows:

(1) Use the FDM algorithm to map the firewall rules into one or more multidimensional rectilinear polygons.

(2) Divide each multidimensional polygon into a set of grid cells, so that these grid cells can be merged more conveniently in the heuristic algorithm.

(3) Propose a heuristic algorithm based on a "greedy" strategy. The core idea of the algorithm is to define a multidimensional rectangle which covers as many grid cells as possible in each step.

Step 1. Mark all grid cells white, and determine a certain dimension F_i of the multidimensional polygon.

Step 2. Choose a grid cell c with the minimum coordinate value in F_i. When there is a grid cell with the right adjacent spatial relation with c, then merge as many grid cells as possible in this dimension to form a RECtangle (REC). Record REC (which corresponds to a rule) and mark it gray. Then the convex part of the gray area in the polygon is determined to be marked as black.

Step 3. Remove the black area from the multidimensional polygon, and then continue with Step 2 in the remaining multidimensional polygons. When all the polygons are marked in gray, output the rectangles and corresponding rules.

The heuristic algorithm FPC possesses the following two properties:

Property 3. In Step 2, REC always covers the region starting from the convex, and the region contains the most grid cells.

Property 4. The previous REC coverage process does not result in a situation whereby the region in the remaining multidimensional polygons, which can be covered by a REC, has to be covered by multiple RECs.

Property 3 is obvious. We prove Property 4 as follows:

Proof: Suppose k-multidimensional rectangles are needed to cover the original multidimensional polygon P. For example, P comprises five grid cells: c_1, c_2, c_3, c_4, and c_5, as shown in Fig. 5. According to the algorithm implementation process, each time the REC covers, the area that contains the initial grid cell and covers the most grid cells is selected. Then, in the multidimensional rectangles c_1, c_3, and c_5 which are formed in the REC coverage, we remove the convex region (c_1 and c_5, which are represented by the gradient-filled area in Fig. 5). At this point, the problem is equivalent to proving that after the convex region is removed, the remaining polygon P' can be covered by a maximum of $k-1$ multidimensional rectangles. In a multi-dimensional polygon P, at least one multidimensional rectangle is needed to cover the convex region. Moreover, based on the property of the MNC problem, the gray region in Fig. 5 allows repeated coverage; therefore, the rest regions of a multidimensional polygon can be covered by up to $k-1$ multidimensional rectangles.

According to the properties of FPC algorithm, in the algorithm implementation process, the REC always covers the region that contains the most grid cells. The algorithm embodies the greedy idea and achieves the local optimal coverage.

3.2.3 Steps and analyses of FPC algorithm

Our algorithm comprises the following four steps:

(1) Map the firewall rules f to an FDM;

(2) Divide the rectilinear polygon into grid cells;

(3) Perform FPC on the multidimensional polygon;

(4) Generate the resulting firewall rules f' from the derived rectangles.

Step 1: Map the firewall rules f to an FDM.

Given a firewall f, we map the rules into a multidimensional matrix to form a set of unit spaces. The procedure of mapping a firewall rule r_i to a k-dimensional matrix M_k is presented in Ref. [2]. Then, we derive the independent unit space sets, each of which can be used to constitute a multidimensional rectilinear

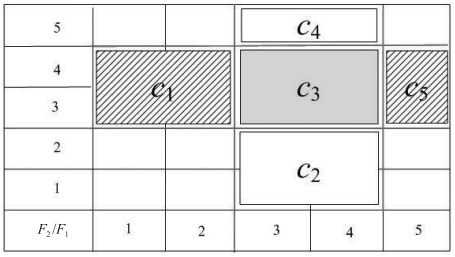


Fig. 5 An example of rectangle covering process.

polygon. Suppose the number of rules is n, the time complexity is $O(kn)$, where k is the dimension.

Let us consider an intuitive example to show the process of this approach, supposing there are five original user specified rules.

Given the firewall f shown in Fig. 6 as input, we can obtain three unit spaces $[(0,4)(3,4)]$, $[(2,2)(1,2)]$, and $[(3,2)(1,3)]$ after the mapping procedure, which can constitute a rectilinear polygon, as shown in Fig. 7a.

Step 2: Divide the rectilinear polygon into grid cells.

In this step, the multidimensional rectilinear polygon (which consists of unit spaces) is divided into grid cells. For any two unit spaces u and v, if they satisfy these two conditions: (1) in dimension F_x, $R(u, v, F_x) =$ *adjacent*, and (2) in any other dimension F_y, $R(u, v, F_y) =$ *included* or $R(u, v, F_y) =$ *crossed*, we let the coordinate value of u in dimension F_y to be the secant, then cut u into two or three sub-unit spaces. This operation on unit spaces is iteratively conducted in a certain dimension until all the k dimensions are completed.

In the worst case, n unit spaces need to be cut for $(k-1)n(n-1)$ times in a k-dimensional space. The time complexity is $O(k\,n(n-1))$, and the number of grid cells is $n + (k-1)n(n-1)$. Here n is the number of unit spaces. However, when all the unit spaces are independent of each other in k dimensions, the number of unit spaces is the same as the number of grid cells.

$$
\begin{aligned}
&r_1 : F_1 \in [0,4] \wedge F_2 \in [5,5] \rightarrow \textit{accept};\\
&r_2 : F_1 \in [0,2] \wedge F_2 \in [2,4] \rightarrow \textit{accept};\\
&r_3 : F_1 \in [5,5] \wedge F_2 \in [3,5] \rightarrow \textit{discard};\\
&r_4 : F_1 \in [0,2] \wedge F_2 \in [0,1] \rightarrow \textit{discard};\\
&r_5 : F_1 \in [2,5] \wedge F_2 \in [0,4] \rightarrow \textit{accept}.
\end{aligned}
$$

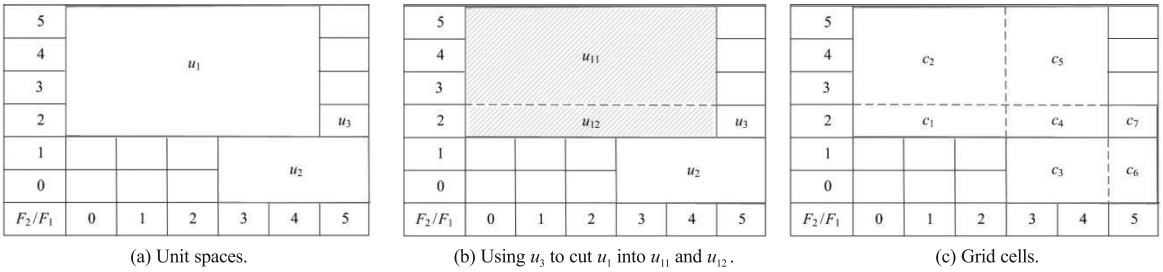
Fig. 6 An example of firewall f with five rules.

Figure 7 illustrates an example of this step. Given unit spaces u_1, u_2, and u_3 as the input, since $R(u_3, u_1, F_1) =$ *adjacent*, and $R(u_3, u_1, F_2) =$ *included*, we use the upper coordinate of u_3 in dimension F_2 as the secant to cut u_1 into two sub-unit spaces u_{11} and u_{12}, which is represented by the gradient-filled area in Fig. 7b. Then this cutting operation is iteratively conducted on all the grid cells, and finally, we obtain seven grid cells from c_1 to c_7 (as shown in Fig. 7c). Here all the grid cells have the same degree of dimensions.

Step 3: Perform FPC on the multidimensional polygon.

(1) To conveniently cover the grid cells, we designed a directed graph structure on the spatial relations among these grid cells. To describe the spatial relations, we consider each grid cell as a vertex in the digraph. If two grid cells are adjacent in a certain dimension, we use a directed line segment which connects these two vertices to describe their adjacency relationship.

Taking the grid cells in Fig. 7c as an example, $R(c_1, c_4, F_1) =$ *adjacent*, $R(c_4, c_7, F_1) =$ *adjacent*, and $R(c_1, c_2, F_2) =$ *adjacent*, their spatial relations can be represented as the digraph as shown in Fig. 8. Here, Dim F_1 and Dim F_2 are the abbreviations of dimensions F_1 and F_2, respectively. For a k-dimensional case, we can



(a) Unit spaces. (b) Using u_3 to cut u_1 into u_{11} and u_{12}. (c) Grid cells.

Fig. 7 An intuitive example of cutting unit spaces into grid cells.

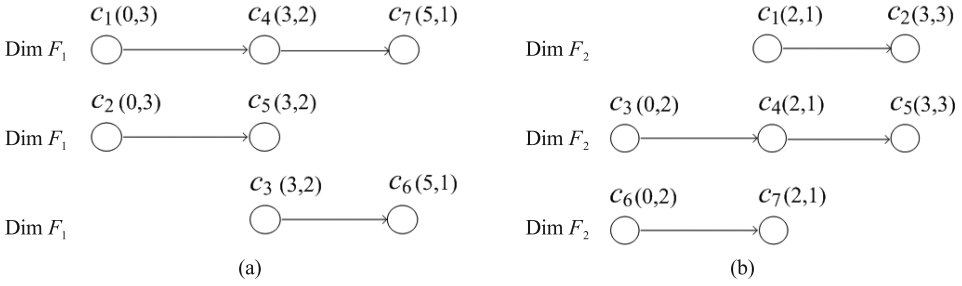


Fig. 8 An example of digraph for the spatial relations of the grid cells in Dim F_1 and Dim F_2, respectively.

construct k digraphs, and in each digraph, each vertex only appears only once.

(2) Based on this digraph, we first choose an initial grid cell. Suppose there are k dimensions from F_1 to F_k, we list all the grid cells which has the minimum coordinate value in the first dimension F_1. Then, we search these grid cells for the one that has the minimum coordinate value in the second dimension F_2. The search operation is iteratively conducted in all the k dimensions until the initial grid cell c is found. Let N be the number of grid cells; then, the time complexity of choosing the initial grid cell is $O(N\log N)$.

For simplicity, we illustrate this method using a two-dimensional case. As shown in Fig. 8a, c_1 and c_2 have the smallest coordinate value in Dim F_1 among all the grid cells. In addition, the coordinate value of c_1 is smaller than that of c_2 in Dim F_2, as shown in Fig. 8b. Therefore, c_1 is the initial grid cell.

(3) First, we search for the linked list l_k whose first vertex is c in the k-th digraph; suppose there are l_k vertex $(c_1 - c_{l_k})$ in the list, then in the $(k-1)$-th digraph, we traverse all the l_k linked lists whose starting points are from c_1 to c_{l_k}. Let l_k be the length, and let the shortest chain length of the traversed l_k linked lists be the width of the rectangle, we can obtain a two-dimensional rectangle. The operation is iteratively conducted from the k-th digraph to the first digraph until a k-dimensional hyper-rectangle is obtained. Then, we remove all the convex vertexes c_v from the k digraphs. In a certain dimension Dim F_i, we can identify that the grid cells constitute convex vertexes in the k-dimensional space, and remove them from the k digraphs if they satisfy the following two conditions: (1) Each grid cell is the starting or the ending vertex of the linked list in this dimension F_i; (2) One or more $(k-1)$-dimensional convex polygons in the other dimensions are exclusively composed of these grid cells. The above operation is iteratively conducted until all the vertices are removed.

Let N be the number of grid cells, to compute the time complexity. We first give the following observations:

(1) In the k dimensions, all grid cells can form k digraphs, and in each digraph, each vertex appears only once.

(2) In each round of covering process, we start from the initial grid cell. Then, we separately perform the traversal operation on the k dimensions to find the largest multidimensional rectangle. Because each grid cell is covered at most once in each round, the traversal operation does not exceed the number of grid cells that needs to be covered in this round.

(3) In each round of covering process, we remove the convex vertices after the multidimensional rectangles are formed. Because in each round, more than one convex vertex will be removed. The number of grid cells required to be covered in the t round is not more than $N - t + 1$, where $1 \leqslant t \leqslant N$.

According to the above consideration, the time complexity of grid cells covering in the worst case is $\sum_{t=1}^{N}(N-t+1)$, that is $O(N^2)$, while in the best case, the grid cells are independent of each other, and the time complexity is $O(N)$.

Taking Fig. 8 as an example, since c_1 is the initial grid cell, we first search with the initial grid cell c_1 for the linked list and find c_1-c_2 (Fig. 8b). Then, we use c_1-c_2 as the edge of rectangle and extend the edge to the maximum in Dim F_1; then, we can obtain the rectangle c_1-c_2-c_4-c_5, as shown in Fig. 8a. Since c_1 and c_2 are the starting points of simultaneously linked lists in Dim F_1, and they constitute a linked list which does not contain any other vertex in Dim F_2, it means c_1 and c_2 are convex grid cells in Dim F_1. Similarly, c_5 is a convex grid cell in Dim F_2. After all the convex grid cells have been removed, we find the initial grid cell c_3 among the rest grid cells. The covering operation is recursively conducted until we get the second rectangle c_3-c_4-c_6-c_7. At this time, these four grid cells c_3, c_4, c_6, and c_7 are all convex. When we removed them, no any grid cell is left, and the algorithm ends. According to the above implementation process, we can finally obtain two rectangles c_1-c_2-c_4-c_5 and c_3-c_4-c_6-c_7 after two rounds of covering operation.

Figure 9 shows the corresponding rectilinear polygon covering process. Given the polygon P shown in Fig. 7c as input, it has seven grid cells from c_1 to c_7. In the first round of covering, c_1 is the initial grid cell, and the maximal rectangle c_1-c_2-c_4-c_5 is added to the cover T. Here, the gradient-filled area represents the already covered area, as shown in Fig. 9a. Next, we remove the convex grid cells c_1, c_2, and c_5, which are represented by the shaded area shown in Fig. 9b. Now the contracted polygon in Fig. 9b has the initial grid cell c_3, and the rectangle c_3-c_4-c_6-c_7 is added to T in this iteration, as shown in Fig. 9c. Finally, following the contraction, all the grid cells are removed and the algorithm terminates normally, as shown in Fig. 9d.

Step 4: Generate the resulting firewall rules f' from the derived rectangles.

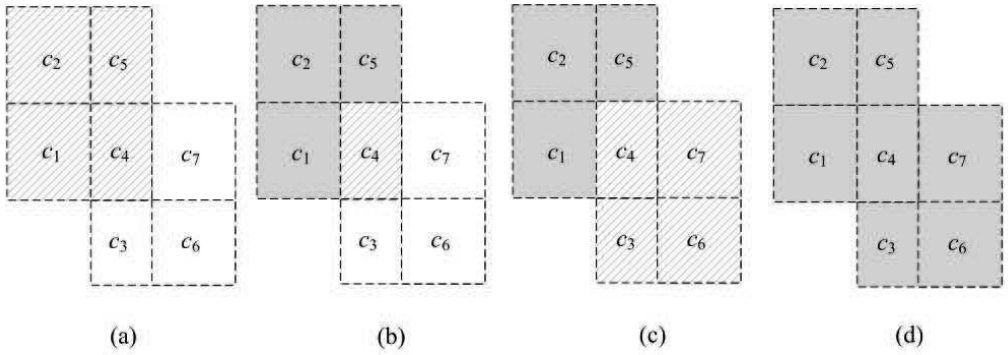


Fig. 9 An example of rectilinear polygon covering in the two-dimensional space. (a) To cover c_1, c_2, c_3, and c_4 with c_1 as the initial grid cells; (b) Remove the convex grid cells c_1, c_2, and c_5; (c) To cover c_3, c_4, c_6, and c_7 with c_3 as the initial grid cells; and (d) Remove all the convex grid cells.

In this step, the resulting firewall rules are generated from the derived rectangles. We first calculate the unit spaces which are formed by those rectangles. Then, we sort them in a descending order according to their regional sizes. The corresponding ordered firewall rules are generated from these ordered unit spaces. Finally, since all the unmarked regions denote the discard condition, a default firewall rule "$F_1 \in D(F_1) \wedge F_2 \in D(F_2) \wedge \cdots \wedge F_k \in D(F_k) \rightarrow$ *discard*" is fixed for the last rule to ensure the completeness of the firewall rules.

For $\mathrm{REC}_1 = \{c_1, c_2, c_4, c_5\}$ and $\mathrm{REC}_2 = \{c_3, c_4, c_6, c_7\}$, we can obtain the corresponding unit spaces: $\{[(0,4)(3,4)], [(2,2)(3,3)]\}$. Based on our rules-generating algorithm, the resulting firewall rules are r_1: $F_1 \in [0,2] \wedge F_2 \in [4,7] \rightarrow$ *accept*, r_2: $F_1 \in [2,4] \wedge F_2 \in [2,4] \rightarrow$ *accept*, and the default rule r_3: $F_1 \in [0,5] \wedge F_2 \in [0,5] \rightarrow$ *discard*.

In this step, we calculate the regional sizes of those hyper-rectangles, and then sort them in a descending order. Let N be the number of grid cells; then, in the worst case, the number of hyper-rectangles is N, and the time complexity is $O((k + \log N)N)$.

Through comprehensive analysis and evaluation, we can derive that the time complexity of our algorithm as $O(T_{\text{best}}) = O(ckn + kn(n-1) + kn^2 + (k + \log n)n) = O(kn^2)$ in the best case and $O(T_{\text{worst}}) = O(ckn + kn(n-1) + kn^4 + (k + 2\log n)n^2) = O(kn^4)$ in the worst case.

4 Experimental Results

In this section, we evaluate the effectiveness and efficiency of the FPC algorithm.

4.1 Effectiveness

Theorem 1: The proposed FPC heuristic approach can certainly reduce the number of rules with a higher degree compared to the FDM method.

Proof: As defined in Ref. [2], the FDM method can be seen as the MNC problem of a multidimensional rectilinear polygon, while the FPC approach can be regarded as an MOC problem in a multidimensional space. For FDM, the resulting number of compressed rules is |MNC|, while the resluting number of rules in FPC is |MOC|. As proved in Ref. [4], $|\mathrm{MOC}| < |\mathrm{MNC}| \leqslant 2|\mathrm{MOC}| - 1$. This result demonstrated that by using an MOC versus an MNC, we can reduce the number of rectangles in the cover by a factor of at most two.

(1) Compression ratios. To evaluate the effectiveness of the firewall rules compression algorithm, we first define the compression ratio metric. Given a firewall f, we use FPC to represent our proposed FPC approach. Let $\mathrm{FPC}(f)$ be the firewall produced by applying algorithm FPC, and $|f|$ be the number of rules in firewall f. The compression ratio of FPC over f is $\dfrac{|\mathrm{FPC}(f)|}{|f|}$.

(2) Experimental data. Firewall rules are considered confidential because of various security concerns, thus, it is not easy to get many real-life firewall rules for evaluation. To circumvent this issue and effectively evaluate the performance of the proposed approach, we generated a set of 50 synthetic firewall rules, comprising 20 to 5000 rules. We divided this set into three smaller sets based on the number of rules. The small set contains the 20 smallest classifiers, and the middle set contains the next 15 larger classifiers, while, the classifiers in the large set all have at least 500 rules, with the largest having 5000 rules. The predicate of each rule has five fields: source IP, destination IP, source port, destination port, and protocol type. We generated these rules by ClassBench[15], which is a well-known benchmark that provides classifiers similar

to real classifiers used in Internet routers and inputs traces corresponding to the classifiers.

(3) Methodology. In our experiments, the rules set was divided into five groups based on sizes, and each group contained eight firewall rules with different rule numbers. We addressed each rule in the five groups by using the FDM, Diplomat, and FPC schemes. The average value was obtained and considered as a performance metric.

As shown in Fig. 10, the compression ratios of FDM, Diplomat, and FPC are all slightly around the average values, which indicates that the compression results are fairly stable. Note that the time complexity of Firewall Decision Diagram (FDD) construction is $O(n^k)$[16]. For the typical five-dimensional firewall rules, k is 5. As a result, the time complexity of Diplomat, which includes the steps of FDD construction, difference resolution, etc., is not less than $O(n^k)$. When the number of rules were more than 1000, the algorithm execution time was over 200 h. Therefore, in this study, no more than 1000 rules were tested for Diplomat.

The mean compression ratios for FDM, FPC, and Diplomat on each set of classifiers (Table 2) show that FDM offers little improvement over Diplomat, and in many cases they provide equally good solutions. However, FPC outperformed them on most of the 50 classifiers. Specifically, on the small and medium sets, FPC resulted in improvements of 8.06% and 6.17% over Diplomat. On the large set, FPC moderately outperformed FDM and Diplomat.

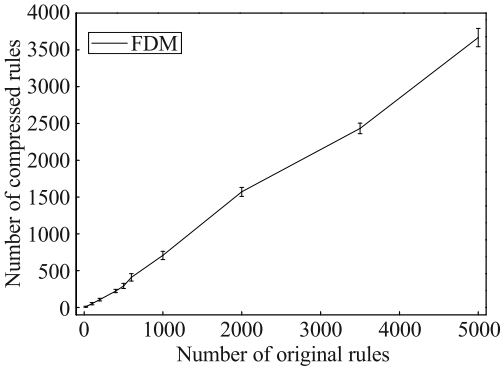
Table 2 Compression ratios of firewall rules.

Rules set	FDM (%)	Diplomat (%)	FPC (%)
Small (20–80)	51.13	54.94	46.88
Medium (100–400)	53.83	55.5	49.33
Large (500–5000)	63.47	63.0	61.15
Mean	56.14	57.81	52.45

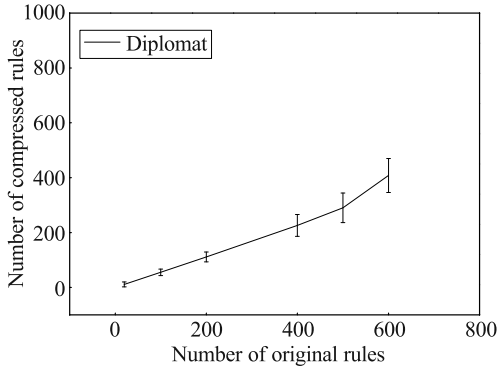
In general, if intersection conditions exist as shown in Fig. 11, the compression ratio of FPC would be better than those of FDM and Diplomat, and the resulting compressed rules would be conflict-free.

4.2 Efficiency

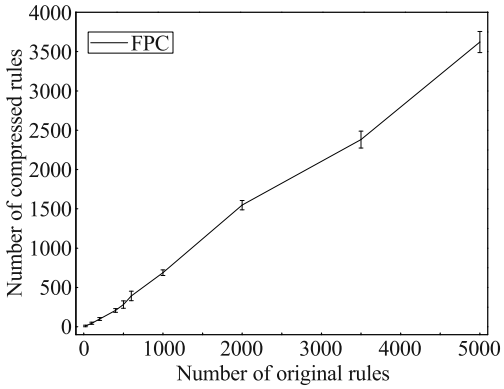
The proposed algorithms were implemented in Java JDK1.6, and we carried out experiments on a server running Linux with Intel six CPU 2.0 GHz. The running



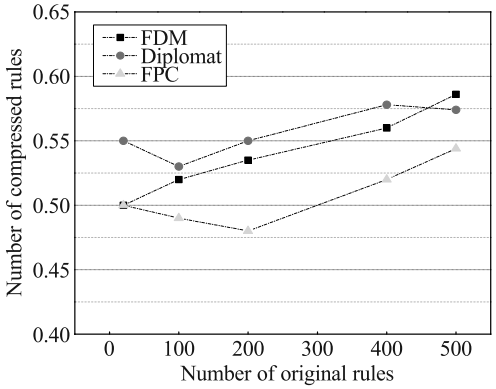
(a) Compression results of FDM.



(b) Compression results of Diplomat.



(c) Compression results of FPC.



(d) Compression ratio of FDM, Diplomat, and FPC.

Fig. 10 Performance comparison of FDM, Diplomat, and FPC.

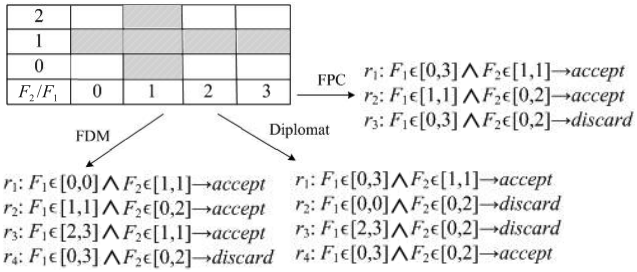
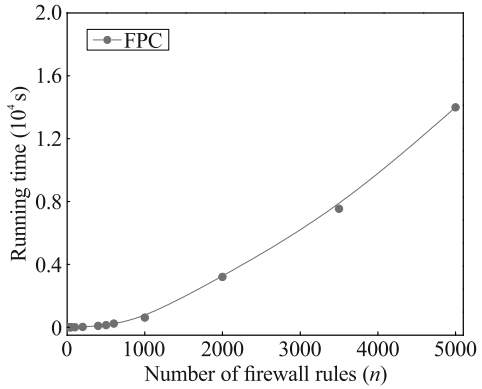
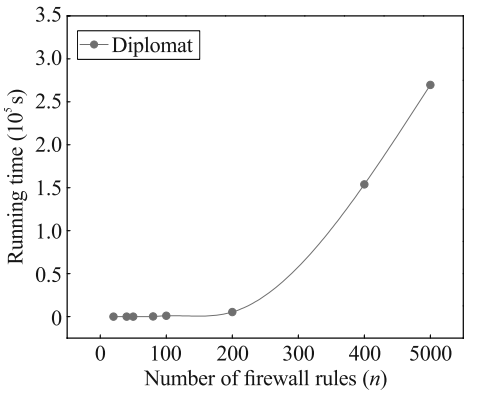


Fig. 11 Rules compression example of FDM, FPC, and Diplomat.

times of FPC and Diplomat (Fig. 12) show that FPC executes more efficiently than Diplomat. Its execution time is within a second on small rules sets, which contain dozens of rules, between a second and a minute on the medium sets, which contain hundreds of rules, and up to a few hours for some large sets containing thousands of rules. As stated above, the time complexity of FPC in the worst case is $O(n^4)$. In fact, the execution time is close to the best case of $O(n^2)$; this is because most of the unit spaces maintain disjunct spatial relations in the multidimensional space. For Diplomat, the time complexity is $O(n^5)$; when the



(a) Running time of FPC



(b) Running time of Diplomat

Fig. 12 Running time with different sizes of firewall rules.

number of rules increases, its execution time increases dramatically.

5 Conclusion

In this paper, we propose a novel scheme FPC to compress firewall rules into conflict-free rules semantically identical to the original rules. First, a geometric model is constructed to transform the firewall rules compression problem into an MOC problem in a multidimensional space, and then, the multidimensional MOC problem is solved using the proposed FPC scheme. We conducted extensive experiments to evaluate the performance of the proposed method. The experimental results show that FPC outperforms FDM and Diplomat in terms of compression ratios. Most of the time, FPC resulted in a high compression ratio while maintaining conflict-free firewall rules. In our future work, we will develop software-defined networking applications[17] that utilize our compression method, and evaluate the feasibility of our method with corresponding experiments. In addition, we will apply powerful techniques studied in computer science, such as parameterized method[18–21], to our method.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Nos. 61672543 and 61402542), Research Foundation of the Education Department of Hunan Province (No. 17B022), and Hunan Provincial Innovation Foundation for Postgraduate (No. CX2014B081).

References

[1] C. R. Meiners, A. X. Liu, and E. Torng, Bit weaving: A non-prefix approach to compressing packet classifiers in tcams, *IEEE/ACM Transactions on Networking (ToN)*, vol. 20, no. 2, pp. 488–500, 2012.

[2] Y. Cheng, W. Wang, G. Min, and J. Wang, A new approach to designing firewall based on multidimensional matrix, *Concurrency and Computation: Practice and Experience* vol. 27, no. 12, pp. 3075–3088, 2015.

[3] D. A. Divekar and R. I. Dowell, Corner stitching: A data-structuring technique for VLSI layout tools, *IEEE Transactions on Computer-Aided Design*, vol. 3, no. 1, p. 87, 1984.

[4] S. Y. Wu and S. Sahni, Covering rectilinear polygons by rectangles, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 9, no. 4, pp. 377–388, 1990.

[5] D. A. Applegate, G. Calinescu, D. S. Johnson, H. Karloff, K. Ligett, and J. Wang, Compressing rectilinear pictures and minimizing access control lists, in *Proc. 18th Ann.*

ACM-SIAM Symp. Discrete Algorithms, New Orleans, LA, USA, 2007, pp. 1066–1075.

[6] A. X. Liu, E. Torng, and C. R. Meiners, Firewall compressor: An algorithm for minimizing firewall policies, in *Proc. 27th Conf. Computer Communications*, Phoenix, AZ, USA, 2008, pp. 176–180.

[7] J. Daly, A. X. Liu, and E. Torng, A difference resolution approach to compressing access control lists, *IEEE/ACM Transactions on Networking*, vol. 24, no. 1, pp. 610–623, 2016.

[8] R. P. Draves, C. King, S. Venkatachary, and B. D. Zill, Constructing optimal IP routing tables, in *Proc. 18th Ann. Joint Conf. IEEE Computer and Communications Societies*, New York, NY, USA, 1999, pp. 88–97.

[9] J. Wang, P. Tan, J. Yao, Q. Feng, and J. Chen, On the minimum link-length rectilinear spanning path problem: Complexity and algorithms, *IEEE Transactions on Computers*, vol. 63, no. 12, pp. 3092–3100, 2014.

[10] V. S. A. Kumar and H. Ramesh, Covering rectilinear polygons with axis-parallel rectangles, in *Proc. 31st Annu. ACM Symp. Theory of Computing*, Atlanta, GA, USA, 1999, pp. 445–454.

[11] P. Berman and B. DasGupta, Complexities of efficient solutions of rectilinear polygon cover problems, *Algorithmica*, vol. 17, no. 4, pp. 331–356, 1997.

[12] W. Liou, J. M. Tan, and R. C. Lee, Minimum rectangular partition problem for simple rectilinear polygons, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 9, no. 7, pp. 720–733, 1990.

[13] R. M. Karp and A. Wigderson, A fast parallel algorithm for the maximal independent set problem, *J. ACM*, vol. 32, no. 4, pp. 762–773, 1985.

[14] S. Suri, T. Sandholm, and P. Warkhede, Compressing twodimensional routing tables, *Algorithmica*, vol. 35, no. 4, pp. 287–300, 2003.

[15] D. E. Taylor and J. S. Turner, Classbench: A packet classification benchmark, *IEEE/ACM Transactions on Networking (ToN)*, vol. 15, no. 3, pp. 499–511, 2007.

[16] A. X. Liu and M. G. Gouda, Diverse firewall design, *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 9, pp. 1237–1251, 2008.

[17] M. Wang, J. Liu, J. Mao, H. Cheng, J. Chen, and C. Qi, Routeguardian: Constructing secure routing paths in software-defined networking, *Tsinghua Science and Technology*, vol. 22, no. 4, pp. 400–412, 2017.

[18] X. Ye, Privacy preserving and delegated access control for cloud applications, *Tsinghua Science and Technology*, vol. 21, no. 1, pp. 40–54, 2016.

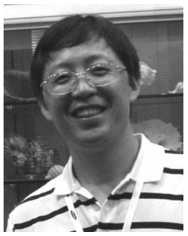
[19] W. Li, Y. Cao, J. Chen, and J. Wang, Deeper local search for parameterized and approximation algorithms for maximum internal spanning tree, *Information and Computation*, vol. 252, pp. 187–200, 2017.

[20] J. E. Chen, C. Xu, and J. X. Wang, Dealing with 4-variables by resolution: An improved maxsat algorithm, *Theor. Comput. Sci.*, vol. 670, pp. 33–44, 2017.

[21] J. You, J. Wang, and Y. Cao, Approximate association via dissociation, *Discrete Applied Mathematics*, vol. 219, pp. 202–209, 2017.

Weiping Wang received the BS degree from Southeast University in 1991, and MS and PhD degrees from Central South University in 1994 and 2004, respectively. She joined Central South University in 1994. Currently, she is a full professor and PhD adviser at Central South University. Her research interests include cyber security and privacy, network coding, and anonymous communication. She has published more than 70 papers in referred journals and conference proceedings. She has presided over four National Natural Science Foundation Projects and participated in more than ten other major scientific research projects. Her teaching courses include computer network, network security, and security of network and system.

Yuzhu Cheng received the BS degree from Hunan University of Science and Technology in 2002 and the MS degree from Hunan University in 2005. He is a faculty of Changsha Social Work College and currently working toward the PhD degree with Central South University, Changsha, China. His research interests include network security, privacy protection, and related areas.



Jianxin Wang received the BS and MS degrees from Central South University in 1992 and 1996, respectively, and received the PhD degree from Central South University in 2001. He is a vice dean and a professor in School of Information Science and Engineering at Central South University, China. His current research interests include algorithm analysis and optimization, parameterized algorithm, bioinformatics, and computer network. He has published more than 150 papers in various international journals and refereed conferences. He is a senior member of IEEE.

Haodong Wang is an associate professor in the Department of Electrical Engineering and Computer Science at Cleveland State University. He received the PhD degree in computer science from College of William and Mary. His research interests focus on information assurance in cyber-physical systems, privacy preserving and user access control in sensor networks, efficient information storage, search and retrieval in pervasive computing, and mobile system security and computing.