Increasing the efficiency of packet classifiers with closed descriptions

Elizaveta F. Goncharova[1[0000-0001-8358-9647]] and Sergei O. Kuznetsov[2[0000-0003-3284-9001]]

[1] National Research University Higher School of Economics, Moscow, Russia
{egoncharova,skuznetsov}@hse.ru

Abstract. Efficient representation of packet classifiers has become a significant challenge due to the rapid growth of data stored and processed in the forwarding, or routing, tables. In our work we propose two algorithms for reducing the size of forwarding tables both in length and width by the deletion of redundant bits and unreachable rules based on FCA analysis. We consider the task of transferring the forwarding packet to the correct destination as the task of multinomial classification. Thus, the process of reducing the forwarding table size corresponds to feature selection procedure with slight modifications. The presented techniques are based on closed descriptions and decision trees. The main challenge in applying decision trees to the task is processing the overlapping rules. To overcome this challenge we propose to employ concept-based hypotheses to delete unreachable actions assigned to the overlapping rules. The experiments were performed on data generated by the ClassBench software. The proposed approach results in significant decrease in bits in the forwarding tables as features.

Keywords: FIB optimization, concept-based hypotheses, decision tree.

1 Introduction and related works

A FIB (forwarding information base) is a wide-spread network instrument used for routing and forwarding packets to the proper output network interface. Due to the rapid growth of the forwarding tables size the time of lookup and forwarding process increases significantly. Modern networking systems require the process of packet transferring to be more and more efficient and fast. In our research we introduce a novel technique for optimization of forwarding tables. Application of the proposed algorithm results in the reduction of the number of bits, which are kept in the memory and used for the lookup process. We consider the task of transferring the forwarding packet to the correct destination in accordance with the FIB as a special task of multinomial classification, where train and test data are the same, so overfitting is not an issue. Thus, the process of reducing the size of the table is considered as the task of feature selection and rule reduction. The presented approaches are based on closed descriptions defined in terms of Formal Concept Analysis (FCA).

Some of the existing techniques for FIB optimization utilize the decision tree approach, e.g. in [1] the authors present a new algorithm using a heuristic based on the

structure built in the classifier. The main idea of algorithm *HiCut* presented in [1] is to create a decision tree based on structural properties of the classifier, where a leaf node stores just a few numbers of rules. In [2] the authors introduce a new algorithm called *HyperCut*, which is the modification of *HiCut*. Each node in the decision tree of *HyperCut* represents a *k*-dimensional hypercube. In comparison to the previous version of the algorithm the authors claim to attain 2 to 10 times memory reduction. The main problem of these approaches is processing the overlapping rules.

In [3] a novel algorithm that reveals the structural properties of FIB is proposed. The authors present a technique for reducing the number of fields (columns) in the forwarding table. The approach proposed in the article is similar to the greedy technique of feature selection. The authors are trying to reduce the rules width by selecting the fields and bits which are important for the classification process. They achieve it by sequential deletion of each field checking whether the classifier keeps the property of order-independence. We will use this algorithm as the baseline in our experiments.

The paper is organized as follows. In Section 2 we describe data and formalize the model of forwarding tables. Section 3 contains the description of the evolving approaches. Experimental results are reported in Section 4. Section 5 concludes the paper.

2 Model description

The basic scheme of packet classification using forwarding table can be presented as follows. The incoming packet goes through the table, and the first row that matches the packet description defines the respective action.

We start with the main definitions of packet forwarding. The table entry is a packet header $H = (h_1, h_2, \dots, h_n)$, $h_i \in \{0,1\}$, which is a sequence of n bits, each of them can take values zero or one. This sequence goes through the ordered set of rules $R = (r_1, r_2, \dots, r_m)$, where each rule is represented by the ordered set of m ternary values 0, 1, and * ("don't care"), and the corresponding action A_j, $j = \overline{1,N}$, where N is a number of all possible actions. This set of rules is often implemented in ternary content-addressable memory (TCAM). The forwarding process looks for exact values for all fields, assigning the packet header to the corresponding action. A header H matches a rule r_i if for every bit from H the corresponding bits from r_i takes either the same or * value [4].

The forwarding table is used with due account of the priority relation on actions. Let $P(A_i)$ be a priority of action A_i, then $P(A_i) > P(A_j)$ if $i < j$. If an input packet matches more than one rule, then the rule with the action having the highest priority is applied.

The initial packet is not given in the binary form. Packet descriptions consist of several fields, the number of fields depends on the specific protocol version (e.g., IPv4 or IPv6). In general, the source and destination hosts, port numbers, or the port numbers range make the fields of classification rule. To follow the definition mentioned above each of the field values should be performed in TCAM format. For in-

stance, the IP-address with the mask can be presented in 32-bit form, where the mask marks the significant bits. Table 1 gives an example of simplified routing table.

Table 1. Example of simplified forwarding table.

IP-address of source port	IP-address of destination port
@145.125.157.1/32	40.140.16.190/32
@195.33.215.197/32	79.205.27.10/32
@195.33.215.196/32	79.205.31.157/32

In this simplified table the rule r_i is built upon one field, which is IP-address of the source port, and the action is represented by IP-address of the destination port.

First, the initial data is transformed to TCAM format, where each number is encoded by zero, one, or * ("don't care") value. Table 2 gives an example of ternary forwarding table, where only last eight bits of the IP-address are encoded. In this example there are 7 various actions A_j and 8 features b_i, which generate the rules r_k of the forwarding table.

Table 2. An example of ternary forwarding table.

	b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7	Action
r_0	0	0	0	0	1	0	0	1	A0
r_1	1	1	0	0	0	1	0	*	A1
r_2	1	1	0	0	0	1	0	0	A2
r_3	0	0	1	1	1	1	1	1	A3
r_4	0	1	0	0	1	0	0	1	A4
r_5	0	1	0	1	1	0	1	1	A4
r_6	0	1	0	1	1	1	0	0	A5
r_7	1	0	1	1	0	0	0	1	A5
r_8	0	0	0	0	1	1	0	1	A6
r_9	0	0	0	1	1	0	0	0	A7

2.1 Data representation

The algorithms we describe below are formulated in terms of Formal Concept Analysis (FCA). To operate with TCAM data we propose a specific pattern structure [5] $(G, (D,\sqcap), \delta)$, where G is a set of objects, D is a set of all possible object descriptions, and $(D,\sqcap)$ is a meet-semi-lattice of object descriptions. Mapping $\delta: G \to D$ takes an object g to its description $d \in (D,\sqcap)$. Galois connection between $(2^G, \subseteq)$ and $(D, \sqsubseteq)$ is defined as follows [6].

$$A^{\square} = \prod_{g \in A} \delta(g), A \subseteq G,$$

$$d^{\square} = \{g \in G | d \sqsubseteq \delta(g)\} \text{ for } d \in (D,\sqcap), \text{where } d \sqsubseteq \delta(g) \Leftrightarrow d \sqcap \delta(g) = d.$$

In our case $G = R$ is a set of rules, D is a set containing all possible TCAM descriptions of each rule in the alphabet $\{0, 1, *\}$, so the pattern structure is $(R, (D, \sqcap), \delta)$. The scheme of intersection operation $\sqcap$ is presented in Table 3.

Table 3. The scheme of intersection operation $\sqcap$.

$\sqcap$	0	1	*
0	0	*	*
1	*	1	*
*	*	*	*

For example, for rules $\boldsymbol{r_4}$ and $\boldsymbol{r_5}$ the result of intersection operation is the following:

$$\delta(r_{4new}) = \delta(r_4) \sqcap \delta(r_5) = \{010 * 10 * 1\}, \text{ and}$$

$$\delta(r_{4new}) \sqsubseteq \delta(r_5), \text{ as } \delta(r_{4new}) \sqcap \delta(r_5) = \delta(r_{4new}).$$

3 Optimization algorithm

We consider the general task as a standard multinomial classification problem, where the rows of the table stay for objects described by features and assigned to the corresponding classes (actions). The application of informative feature selection results in revealing the minimal combination of the informative features, thus decreasing the width of the routing table. Therefore, the look-up procedure of assigning the packet to the corresponding action can become faster. We consider two techniques based on concept-based hypotheses. The first approach is based on a variation of Close-by-One (CbO) algorithm [7]. This method results in constructing a minimal feature subset that determines the corresponding action and the reduction in the number of rules. The second approach combines concept-based hypotheses as a preprocessing step for deleting the overlapping rules with the decision tree algorithm for revealing the informative features.

3.1 Concept-based hypotheses

Concept-based hypotheses [8] used to generate rules with short premises are reformulation of JSM-hypotheses [9] in terms of formal concepts. Data can be represented by N contexts describing each of N actions (classification results) $K_i = (R_i, (D, \sqcap), \delta_i)$, $i = \overline{0, N-1}$, where R_i is a set of the i-th action examples; mapping $\delta_i: R_i \to D$ assigns an i-action example g_i to description $d \in (D, \sqcap)$, $i = \overline{0, N-1}$. The derivation operators in these contexts are defined by superscripts $\square$. Thus, the intent of i-th action examples are denoted by $r_i^{\square}$. Intents of context K_i are called i-th action intents.

3.2 Method based on Close-by-One algorithm

The first approach is based on an adaptation of CbO algorithm in the depth-first strategy [5]. The basic scheme of the proposed method is as follows.

For each context $K_i = (R_i, (D,\sqcap), \delta_i),\ i = \overline{0, N-1}$ we build the CbO tree trying to define a minimal feature subset responsible for defining the i-th action. Let R_i^{node} be a set of rules, and $R_i^{node^{\square}}$ be a common description for each rule from R_i^{node}, where $node$ is a node index in CbO tree (e.g. for the root $node$ equals zero, for the root's children nodes node will be one, etc.).

1. The root of the tree is a pair $\left(R_i^0, R_i^{0^{\square}}\right)$, where $R_i^0 = \emptyset$ and $R_i^{0^{\square}} = \emptyset$.
   Its child nodes consist of just one rule and its description $\left(R_i^1, R_i^{1^{\square}}\right)$, $R_i^1 = \{r_i^1\}$ and $R_i^{1^{\square}} = \delta(r_i^1)\ \forall r_i^1 \in R_i$. If $\delta(r_i^1)^{\square}$ includes a rule r_j^1 corresponding to action $j > i$, then the rule r_j^1 can be deleted from the routing table as unreachable. It is explained by the priority property, because each packet that satisfies r_j^1 also satisfies r_i^1, hence as $P\left(r_j^1\right) < P(r_i^1)$, r_j^1 will never be reached.
2. Having created the children nodes of the first generation, we construct the next generations of children nodes $\left(R_i^{node}, R_i^{node^{\square}}\right)$, $node > 1$. To accomplish this step we add one of the remaining rules to the previous rules set R_i^{node-1}. To get the feature-bit vector describing this new set of rules we should intersect the feature-bit vector corresponding to the added rule $\delta\left(r_i^{node}\right)$ with the current node description $R_i^{(node-1)^{\square}}$. This step can be formulated in accordance with the following rules.

$$R_i^{node} = R_i^{node-1} \cup \left\{r_i^{node}\right\},\ \forall\, r_i^{node} \in R_i \backslash R_i^{node-1};$$

$$R_i^{node^{\square}} = R_i^{node-1^{\square}} \sqcap \delta\left(r_i^{node}\right).$$

3. If $R_i^{node^{\square\square}}$ includes a rule r_j^{node} corresponding to action $j > i$, then we have got an overgeneralized description. We should return to the parent $node - 1$ and add one of the remaining rules. We aim to create the most common description of the i-th action that does not cover the description of other actions.

Example. Consider the work of the method by the example of data in Table 2. As we have only one rule for A_0, we leave it without modification, so $R_0^1 = \{r_0\}$ and $R_0^{1^{\square}} = \{00001001\}$.

The first action is also defined by one rule only: $R_1^1 = \{r_1\}$ and $R_1^{1^{\square}} = \{1100010 *\}$, however, $R_1^1 = \{r_1\}$ and $R_1^{1^{\square\square}} = \delta(r_1)^{\square} = \{r_1, r_2\}$, where r_2 defines A_2. According to the second step of the algorithm, since $P(A_1) > P(A_2)$, rule r_2 can be deleted from the routing table as unreachable.

The fourth action is determined by two rules $\{r_4, r_5\}$. The second generation of children is $R_4^2 = \{r_4, r_5\}$, $R_4^{2^{\square}} = \{010 * 10 * 1\}$. $R_4^{2^{\square\square}} = \{r_4, r_5\}$, which means that $R_4^{2^{\square}}$ is the most compact description for action A_4.

For the fifth action there are also two rules $R_5^2 = \{r_6, r_7\}$, $R_5^{2^{\square}} = \{*** 1 ** 0 *\}$. However, $R_5^{2^{\square\square}} = \{r_6, r_7, r_9\}$, where r_9 defines A_7, in accordance with the third step of

the algorithm, the obtained description $R_5^{2\square}$ is too general, and we should return to the parent nodes $R_5^1 = \{r_6\}$ and $R_5^1 = \{r_7\}$. Thus, action A_5 cannot be presented by one rule, both rules r_6 and r_7 should be kept in the final routing table.

The actions A_3, A_6 and A_7 remain the same, because they are described by one rule only. To illustrate the process described above we present the part of CbO-tree built for FIB given in Table 2 (Fig. 1). The final FIB is given in Table 4.

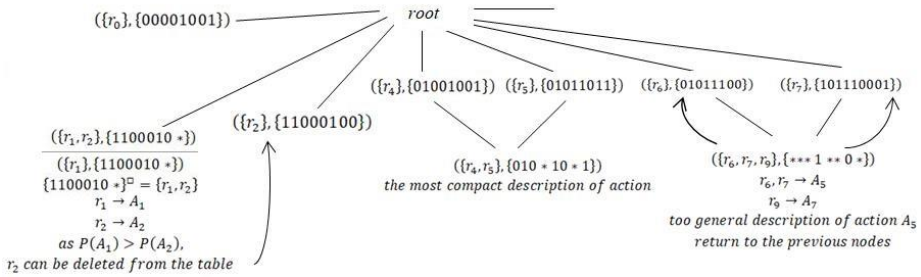

Fig 1. A part of CbO tree.

Table 4. An example of forwarding table reduced with CbO algorithm.

	b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7	Action
r_0	0	0	0	0	1	0	0	1	A0
r_1	1	1	0	0	0	1	0	*	A1
r_3	0	0	1	1	1	1	1	1	A3
r_{4new}	0	1	0	*	1	0	*	1	A4
r_6	0	1	0	1	1	1	0	0	A5
r_7	1	0	1	1	0	0	0	1	A5
r_8	0	0	0	0	1	1	0	1	A6
r_9	0	0	0	1	1	0	0	0	A7

It should be mentioned that the proposed technique does not affect the width of the routing table significantly. It can reduce the number of informative features for each action separately. Besides, it is able to decrease the length of the table by deleting unreachable actions and compressing the number of rules. In some cases the width of the table can also be reduced, for instance, if a feature-bit takes the same value for each rule in the table (i.e. the column of the table consists either of zeros, or ones), this feature-bit can be deleted from the table as uninformative.

3.3 Decision tree and concept-based hypotheses

The second approach uses concept-based hypotheses in a different way. Here we use them not for feature selection, but in the preprocessing step for deleting unreachable rules. While the final stage of feature selection is performed by a standard machine learning technique, decision tree induction in our case.

As we already mentioned some rules can be redundant in the initial table. These unreachable rules complicate the process of building the decision tree, whereas they should not be taken into account in the first way. We consider using concept-based hypotheses to detect them. Having deleted the unreachable rules we generate a decision tree. To create the optimized table we parse the decision tree finding the route for each action with zero error. A route is represented as a row from the optimized table.

As in the algorithm based on CbO we find pairs $\left(R_i^{node}, R_i^{node^{\square}}\right)$, where $R_i^{node^{\square}}$ defines the minimal description for i-th action to detect the unreachable rules. If the proportion of "*" in $R_i^{node^{\square}}$ is less than some threshold ε, and $R_i^{node^{\square\square}}$ contains a rule r_j^{node} corresponding to action $j > i$, then we perform a checking procedure as follows. For all rules $r_i^{node} \in R_i^{node^{\square\square}} \backslash \left\{r_j^{node}\right\}$ if $\delta\left(r_i^{node}\right) \sqsubseteq \delta\left(r_j^{node}\right)$, then r_j^{node} is an unreachable rule and can be deleted from the forwarding table.

Threshold ε is used to catch overgeneralized descriptions $R_i^{node^{\square}}$ that can match large number of rules, we set it to $1/2$ in this work. For instance, in example given in Section 3.2 the proportion of "*" in $R_5^{2^{\square}} = \{*** 1 ** 0 *\}$ equals to $3/4$, which is more than a half. So, we assume that it is an overgeneralized description and there is no need to compute $R_5^{2^{\square\square}}$ and check the inclusion. In this algorithm we do not aim at finding minimal hypotheses for the actions, but at deleting the unreachable rules. Thus, this stage is responsible for decreasing the length of the routing table. We should mention that ε is a hyperparameter aiming at avoiding long execution time, in our experiments the value $1/2$ has provided good performance; however, its impact could be examined more carefully in future works.

Upon deleting all unreachable rules we propose to use decision tree algorithm to find the routes that are able to distinguish all the actions. This stage results in selection of the feature-bits that are informative for classification process, this selection decreases the width of the table. The choice of decision tree algorithm is based upon two reasons:

- built-in procedure of feature selection, thus finding a rule for this or that action we obtain a short way of defining it.
- overfitting does not present a problem for this specific task, because the routing table should be an exact classifier by definition, future data cannot violate it without a general rearrangement of the routing scheme due to external reasons.

We use python implementation of decision tree classifier based on CART algorithm [10] that constructs binary tree structure and information gain for feature selection. It should be mentioned that standard machine learning techniques are not able to operate with pattern structures, therefore, to create a decision tree we encode the features with the following rules:

$$\begin{cases} b_{i0} = 1 \\ b_{i1} = 0 \end{cases}, b_i = 0; \begin{cases} b_{i0} = 0 \\ b_{i1} = 1 \end{cases}, b_i = 1; \begin{cases} b_{i0} = 0 \\ b_{i1} = 0 \end{cases}, b_i = *.$$

This encoding scheme respects the intersection operation given by Table 3. Upon processing the bits can be simply decoded into the initial ternary form.

Having created the decision tree for the initial data without the unreachable rules we can apply the simple false-positive check procedure to check the correctness of the classification results.

Thus, in the proposed method the length of the forwarding table is reduced by applying concept-based hypotheses, whereas, the decision tree with feature selection reduces the width of the table. We have applied this method to the sample FIB given in Table 2.

Example. Let us consider the optimization procedure of the sample FIB given in Table 2 using the proposed method. In this specific example the most pairs $\left(R_i^{node}, R_i^{node^{\square}}\right)$ forming the nodes of CbO tree describe one rule only (see Fig. 1), so they are not included in the procedure of unreachable rules defining. However, there are several pairs that should be processed. The first pair is $\left(R_1^1, R_1^{1^{\square}}\right) = (\{r_1\}, \{1100010\ *\})$, where $R_1^{1^{\square\square}} = \{r_1, r_2\}$ and r_2 corresponds to action A_2, which has less priority than A_1 defined by r_1. Thus, we should check ε for $R_1^{1^{\square}}$; the proportion of "*" in $R_1^{1^{\square}}$ equals to $1/8$, which is less than $\varepsilon = 1/2$. This means that the description is not too general, and r_2 is a candidate for unreachable rule. Then we examine the inclusion of rules' descriptions $\delta\left(r_i^{node}\right) \sqsubseteq \delta\left(r_j^{node}\right)$. In our case

$$\delta(r_1) \sqsubseteq \delta(r_2) \Leftrightarrow \{1100010\ *\} \sqcap \{11000100\} = \{1100010\ *\} = \delta(r_1).$$

This means that r_2 is an unreachable rule and can be deleted from the forwarding table.

The second pair which describes more than one rule is $\left(R_4^2, R_4^{2^{\square}}\right) = (\{r_4, r_5\}, \{010 * 10 * 1\})$. The set $R_4^{2^{\square\square}} = \{r_4, r_5\}$ does not include rules corresponding to different actions (both r_4 and r_5 define action A_4) and, hence, there are no unreachable rules in this pair.

The third candidate is $\left(R_5^2, R_5^{2^{\square}}\right) = (\{r_6, r_7\}, \{***\ 1\ **\ 0\ *\})$, where $R_5^{2^{\square\square}} = \{r_6, r_7, r_9\}$. Rule r_9 corresponds to A_7, while r_6 and r_7 define A_5. However, as has been mentioned above, the proportion of "*" in $R_5^{2^{\square}}$ is greater than a threshold ε. Thus, we assume that the obtained description is too general and there are no unreachable rules in the set $R_5^{2^{\square\square}}$. In this case the assumption is correct, because action A_5 cannot be described by the one rule only, both r_6 and r_7 should be kept in the final table. Neither the description of r_6, nor the description of r_7 covers $\delta(r_9)$, which means that r_9 is a reachable rule.

Application of the preprocessing stage resulted in deleting of one unreachable rule r_2 from the initial sample table. After this step we apply decision tree procedure to generate the paths of bits, which are able to define remaining actions, and build the optimized forwarding table using these paths. The optimized version of FIB given in Table 2 is presented in Tables 5 and 6.

Table 5. An example of reduced forwarding table.

	b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7	Action
r_0	*	0	0	*	*	0	*	1	A0
r_1	*	1	*	*	0	1	*	*	A1
r_3	*	0	1	*	*	*	1	*	A3

$r_{4\text{new}}$	*	1	*	*	*	0	*	*	A4
r_6	*	1	*	*	1	1	*	*	A5
r_7	*	0	1	*	*	*	0	*	A5
r_8	*	0	0	*	*	1	*	1	A6
r_9	*	0	0	*	*	*	*	0	A7

Upon the deletion of unreachable rules the decision tree classifier has revealed the set of uninformative bits $\{b_0, b_3\}$, which are not included in any classification rule. These bits take "*" value for each rule in Table 5.

Table 6. An example of reduced forwarding table without redundant bits.

	b_1	b_2	b_4	b_5	b_6	b_7	Action
r_0	0	0	*	0	*	1	A0
r_1	1	*	0	1	*	*	A1
r_3	0	1	*	*	1	*	A3
r_4	1	*	*	0	*	*	A4
r_5	1	*	1	1	*	*	A5
r_6	0	1	*	*	0	*	A5
r_7	0	0	*	1	*	1	A6
r_8	0	0	*	*	*	0	A7

3.4 False-positive check

If we delete some bits from initial table we may have a so called false positives, when some packet satisfies the reduced table (without several bits), whereas it does not correspond to any rule in the initial FIB. To make the problem clear, let consider two reduced tables (table 4 and 6), obtained with the proposed approaches.

In accordance with the resulting table 4 the packet $h_0 = (010\mathbf{1}10\mathbf{0}1)$ will be forwarded to A_4 by r_{4new} rule, whereas h_0 does not satisfies any of the initial actions r_4 or r_5, which have been the basis for this new rule.

In table 6 the same problem occurs. For example, let $h_0 = (1\mathbf{01}011\mathbf{1})$ be a forwarded packet. In accordance with the values of the 1st, 2nd, and 6th bits the reduced table will assign this packet to action 3, whereas actually this packet should not be assigned to any action and should be stopped by the table.

To prevent this type of errors a false-positive check procedure should be included in the algorithms [3]. The procedure is implemented as follows, if some rules have been modified, then we should keep its initial variant in memory (32 bits and the corresponding action). Thereafter, if some input packet satisfies the new modified rule, then we check whether it also satisfies the initial rules (the ones we keep in the memory). If it suits one of them, the packet should be forwarded to the corresponding action; it is dismissed, otherwise. The process of checking is a simple comparison of two points in multidimensional space. So, the deleted bits are not included in the process of the classification procedure itself, but they are kept to prevent false positives.

4 Experimental results

The experiments were performed using the synthesized data provided with ClassBench software [11]. ClassBench generates sample routing table according to the parameters obtained from the real FIB. The synthesized tables used for the experiments consisted of the IP-address of the source port with 32-bit mask as description and IP-address of the destination port as the output action. We evaluated three generated routing tables characterized by 32 bits and consisting of 100, 500, and 906 rules, respectively.

Two proposed methods were applied to the tables described above. We compared the performance of the proposed methods with the results of the approach similar to the one presented in [3]. The authors of [3] utilize structural properties of FIB and reduce the width of the table by deleting the bits which do not affect the order-independence property. This algorithm is close to greedy technique of feature selection, where the order-independence property is checked instead of the information-gain criterion. This algorithm acts as a baseline in the experiments. The results obtained during the experiments are presented in Tables 7-9, where "Order independence" stay for the approach from [3]. We assess the performance with respect to the following properties.

Reduced number of feature-bits (column 1) shows how many bits of the 32 initial ones have been declared informative. **Reduced number of rules** (column 2) gives the amount of rules in the final table. This property demonstrates how many rules have been declared unreachable or have been united. The last property (column 3) says how many actions have been deleted from the table as unreachable.

Table 7. The results of optimization for the table with 100 rules, 32 bits, and 57 unique actions

Method	Reduced number of features	Reduced number of rules	Number of deleted actions
CbO-based	20	52	2
DT + JSM	14	59	2
Order-independence	10	86	2

Table 8. The results of optimization for the table with 500 rules, 32 bits, and 95 unique actions

Method	Reduced number of features	Reduced number of rules	Number of deleted actions
CbO-based	22	95	32
DT + JSM	15	114	32
Order-independence	29	367	32

Table 9. The results of optimization for the table with 906 rules, 32 bits, and 95 unique actions

Method	Reduced number of features	Reduced number of rules	Number of deleted actions
CbO-based	31	84	38
DT + JSM	30	79	38
Order-independence	29	309	38

We can see that the best results in reduction of table width are obtained by the decision tree algorithm in combination with concept-based hypotheses. Applying concept-based hypotheses resulted in deleting two of 57 in the first experiment, and 32 and 38 actions of 95 in the second and the third experiments respectively. This approach deleted more than a half of all initial features of the first and second synthesized FIBs. In two of three experiments the length of the table was reduced by CbO-based algorithm in the best way. It confirms the fact that the first approach succeeds in deleting redundant rules, while the other techniques are better in width reduction. It should be mentioned that we keep in the memory initial rules, which constitute the new modified rules and correspond to reachable actions, in order to perform false-positive check procedure by necessity.

The baseline approach utilizing order-independence property [3] showed the best results in minimizing the width of a forwarding table in the first experiment with 100 rules and 57 unique actions. However, it should be mentioned that respecting order-independence property one increases the number of rules. Turning the table into order-independent format requires extending of some rules and decoding the “don’t care” value into the zeros and ones in order to prevent conflict of rules.

5 Conclusion

In our work we have presented two approaches to forwarding table minimization based on decision trees and concept-based hypotheses. The first technique is based on CbO-tree construction using a special pattern structure. The second approach utilizes decision tree classification algorithm in combination with concept-based (JSM) hypotheses (DT + JSM) aiming to delete the unreachable rules and reduce the length of the table.

The experiments performed on data provided by the ClassBench software showed that the best trade-off between decreasing the width and the length of the classifier is obtained by DT + JSM technique. This method resulted in significant reduction in both the rules and bits number. The former was obtained by revealing the contradicting hypotheses and, thus, unreachable rules deletion, whereas the latter was achieved by applying the decision tree algorithm to the modified table without unreachable rules. The proposed approaches were compared to the existing technique based on keeping order-independence property of the table. Whereas the number of deleted redundant features is comparable, the number of the rules kept in the final table is

larger for the order-independent approach. The method based on CbO-tree construction resulted in significant reduction of routing table length, which was obtained by intersection of the rules corresponding to specific action; however, it could not reduce big number of features.

Overall, the proposed algorithms can be applied to the task of forwarding table minimization. In this work we overview the simplified version of the table that does not include range features. Thus, in our future research we are planning to apply interval pattern structures to process such type of fields and make our algorithms competitive with the state-of-the-art approaches.

Acknowledgements

The work of Sergei O. Kuznetsov shown in all the sections has been supported by the Russian Science Foundation grant no. 17-11-01276 and performed at St. Petersburg Department of Steklov Mathematical Institute of Russian Academy of Sciences, Russia.

References

1. Gupta, P., McKeown, N., Classifying packets with hierarchical intelligent cuttings. Ieee Micro 20(1), 34-41(2000).
2. Singh, S., Baboescu, F., Varghese, G., Wang, J., Packet classification using multidimensional cutting. In Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, pp. 213-224. ACM (2003).
3. Kogan, K., Nikolenko, S. I., Rottenstreich, O., Culhane, W., Eugster, P., Exploiting order independence for scalable and expressive packet classification. IEEE/ACM Transactions on Networking, vol. 24(2), pp. 1251-1264 (2015).
4. Kogan, K., Nikolenko, S., Eugster, P., Ruan, E., Strategies for mitigating TCAM space bottlenecks. In 2014 IEEE 22nd Annual Symposium on High-Performance Interconnects pp. 25-32. IEEE (2014).
5. Ganter, B. and Kuznetsov, S., Pattern Structures and Their Projections, Proc. 9th Int. Conf. on Conceptual Structures, ICCS'01, G. Stumme and H. Delugach, Eds., Lecture Notes in Artificial Intelligence, vol. 2120, pp. 129-142 (2001).
6. Kaytoue, M., Kuznetsov, S. O., Napoli, A., and Duplessis, S., Mining gene expression data with pattern structures in formal concept analysis. Information Sciences, vol. 181(10), pp. 1989-2001 (2011).
7. Kuznetsov, S. O., Learning of simple conceptual graphs from positive and negative examples. European Conference on Principles of Data Mining and Knowledge Discovery. Springer, Berlin, Heidelberg, 1999, pp. 384-391.
8. Kuznetsov, S. O., Machine learning on the basis of formal concept analysis. Automation and Remote Control, vol. 62(10), pp. 1543-1564 (2001).
9. Finn, V. K., Plausible reasoning in systems of JSM type. Itogi Nauki i Tekhniki, Seriya Informatika, 1991 [in Russian].
10. Breiman, L., Friedman, J. H., Olshen, R. A., Stone, C. J., Classification and regression trees. Belmont, CA: Wadsworth. International Group, 432 (1984).
11. ClassBench: A packet classification benchmark, http://www.arl.wustl.edu/classbench/.