How to deal with range-based packet classifiers

Vitalii Demianiuk
IMDEA Networks Institute
Steklov Institute of Mathematics at St. Petersburg

Kirill Kogan
IMDEA Networks Institute

ABSTRACT

Efficient representations of multi-field packet classifiers with fields represented by ranges is a core mechanism to express services on data plane. To implement classifiers in *ternary-addressable* memory (TCAM), each range should be encoded into multiple ternary bit strings whose number is at most linear to the width (in bits) of a represented field independently from range encoding method. In this paper we introduce a notion of a *subrange* allowing to represent a field range on any chosen subset of bit indices that significantly improve efficiency of classifier representations. Our analytic results are confirmed with a comprehensive evaluation study showing applicability of our approach to implement desired levels of expressiveness and scalability in packet classifiers.

1 INTRODUCTION AND MOTIVATION

Packet classification is a core building block implementing packet processing programs on data plane. With the adoption of OpenFlow [14] and P4 [15], packet classification has become even more prominent. Each *packet classifier* is an ordered set of rules, where a *rule* consists of the filter (matching packet headers) and the associated action to be applied on matched packets. A *filter* is a concatenation of field representations participating during classification; in the simplest form each field in a filter is represented by exact values. In this case packet classifiers can be implemented with a constant lookup time but the number of rules can be infeasible to fit in memory. In the other extreme case, to address the scalability constraint, fields can be represented by ranges of values. In this case the number of rules is significantly reduced but the lookup complexity increases. As a result, some intermediate forms of field representations were introduced as *prefixes* or more general ternary bit strings, where every bit has three values: zero, one, or don't care. For ternary bit strings, a specialized *ternary content addressable* (TCAM) memory was introduced that is actually a coprocessor running multiple searches in parallel [12].

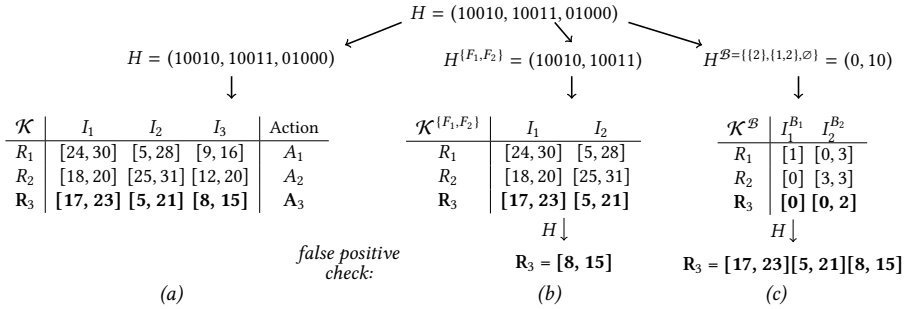
Related work. There are two major directions dealing with range-base packet classifiers. The first one is independent from structural properties of classifiers encoding every field range by multiple prefixes or ternary bit strings whose number is at most linear to the field size (in bits) [3, 4, 11, 16, 17]. Hence, each rule whose fields are represented by ranges is actually a classifier with exponential number of rules based on ternary bit strings. Recently, [1] proposes efficient encoding for the special case of short ranges. Other range encoding methods exploiting structural properties can achieve more compact representations [2, 5] but usually they perform well only when the number of encoded ranges is relatively small. Note that both these lines of research consider transformations to *equivalent* classifiers.

Recently, [6, 9, 10, 13] proposed representations of packet classifiers exploiting their structural properties like *rule disjointness*; observe that these representations become equivalent to originally given classifiers only when the lookup table based on a subset of fields is complemented by the false-positive check on a single matched rule. This allows to balance which fields are required to implement desired structural properties in the lookup table and which are going to the false-positive check where range encoding is unnecessary. It allows to reduce the size of classifier representations. To store classifier in TCAM, a participating subset of ranges in the lookup table should be encoded by one of the previously mentioned methods [3, 4, 11, 16, 17].

Our contributions. In this paper we are going beyond [10] and propose *range reduction* (RR) methods allowing to implement structural properties of classifiers with per-bit resolution (and not with per-field as in [10]), when fields are represented by ranges. We introduce an interesting notion of a *subrange* on a predefined set of range bit-indices allowing to construct equivalent representations of packet classifiers with per-bit resolution avoiding intermediate range expansions. To illustrate this, consider a classifier $\mathcal{K}$ in Fig. 1a based on three range fields. For an incoming header H, the R_3 rule is matched. All three rules are *disjoint* (do not match the same header) on three fields. Note that two fields are sufficient to keep rule disjointness as it is done in [10]. Still these two ranges in every rule are to be encoded by one of the range encoding methods (e.g., prefix expansion [17] or SRGE

SOSR' 19, April 2019, San Jose, California, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn



$\mathcal{K}$	I_1	I_2	I_3	Action
R_1	[24, 30]	[5, 28]	[9, 16]	A_1
R_2	[18, 20]	[25, 31]	[12, 20]	A_2
$\mathbf{R_3}$	**[17, 23]**	**[5, 21]**	**[8, 15]**	$\mathbf{A_3}$

$\mathcal{K}^{\{F_1,F_2\}}$	I_1	I_2
R_1	[24, 30]	[5, 28]
R_2	[18, 20]	[25, 31]
$\mathbf{R_3}$	**[17, 23]**	**[5, 21]**

$\mathcal{K}^{\mathcal{B}}$	$I_1^{B_1}$	$I_2^{B_2}$
R_1	[1]	[0, 3]
R_2	[0]	[3, 3]
$\mathbf{R_3}$	**[0]**	**[0, 2]**

range encoding methods

	prefix [17]		SRGE [3]	
	# entries	# bits	# entries	# bits
$\mathcal{K}$	105	1575	60	900
$\mathcal{K}^{\{F_1,F_2\}}$ [10]	39	390	30	300
$\mathcal{K}^{\mathcal{B}}$, RR	4	12	4	12

Figure 1: Our RR method vs. previous works: (a) original classifier $\mathcal{K}$ whose rules consist of three 5-bit fields; (b) the representation of $\mathcal{K}$ as in [10] exploiting rule disjointness based on two ranges; (c) the proposed RR implementing rule disjointness on three bits; (d) comparison of three different representations based on prefix and SRGE range encoding methods.

based on Gray encoding [3]). Intuitively, representations narrowing down the covered field ranges can significantly improve memory requirements. This is a reason why the notion of subranges introduced in this paper allow to implement rule disjointedness only on 3 bit indices (see Fig. 1c). Fig. 1d demonstrates advantages of per-bit resolution leading to significant reduction in TCAM memory requirements both in total bits and ternary entries (e.g., RR allows to reduce the number of ternary entries maintained in TCAM from 60 to 4 for SRGE range encoding).

Based on properties of subranges, we introduce heuristics to find efficient classifier representations in TCAMs. We demonstrate the viability of our approach through comprehensive evaluation where in the extreme case all six fields are based on ranges. Such classifiers cannot be implemented in TCAM with the conventional methods dealing with equivalent classifiers. The evaluation results together with analytic observations confirm that subranges could be an interesting direction to implement a fundamental tradeoff between expressiveness and scalability.

2 MODEL DESCRIPTION

A packet *header* $H = (H_1, \ldots, H_k)$ is a concatenation of k header fields, where each header field H_i is a binary string on w_i bits; headers are matched by classifiers. A *classifier* $\mathcal{K} = \{R_1, \ldots, R_N\}_{\prec}$ is an ordered (by $\prec$) set of rules, where each *rule* $R_i = (\mathcal{F}_i, A_i)$ consists of a *filter* $\mathcal{F}_i$ and the associated *action* A_i. A filter $\mathcal{F} = (F_1, \ldots, F_k)$ is a sequence of k field representations, where each *field representation* F_i is on w_i bits corresponding to the header field H_i. In this paper we consider two types of field representations: (1) a range of values; and (2) a *ternary* bit string, where each bit can have one of the three values 0, 1, or $*$ ("don't care"). We say that a range I is *matching* a value x if $x \in I$. We say that a ternary bit string T *matching* a binary string H of the same length l if for each bit index $1 \leq i \leq l$ either $H[i] = T[i]$ or $T[i] = *$. In the case of ternary field representations, a filter is a ternary bit string produced by concatenation of the ternary bit strings of the corresponding field representations. A filter $\mathcal{F}$ *matches* a header H if every F_i of $\mathcal{F}$ matches the corresponding header field H_i. We say that two filters $\mathcal{F}_1$ and $\mathcal{F}_2$ are *disjoint* if no single header matches both of them. Otherwise, $\mathcal{F}_1$ and $\mathcal{F}_2$ *intersect*. Two rules intersect (are disjoint) if their filters intersect (are disjoint). The main purpose of the classification process is to find the action corresponding to the highest priority rule matching a given header that we call a *classifying* rule. If there is no rule matching H in $\mathcal{K}$, the classification result is a default action that differs from all actions in $\mathcal{K}$. Two classifiers (or their representations) are *equivalent* if the classification result coincides in both classifiers (representations) for all headers.

Example 2.1. A sample 2-field ($k = 2$) range-based classifier $\mathcal{K}$ with 3-bit fields ($w_1 = w_2 = 3$).

$\mathcal{K}$	I_1	I_2	Action
R_1	[0, 6]	[2, 5]	A_1
R_2	[0, 3]	[1, 3]	A_2

A header $H = (010, 001)$ is classified by R_2 since it is matched only by R_2; a header $H' = (010, 011)$ is matched by both R_1 and R_2 but H' is classified by R_1.

3 RANGE ON A SUBSET OF BITS

Recall that each range is encoded into multiple ternary entries whose number is at most linear to the field width (in bits) independently from range encoding methods [16]. Intuitively, taking values on a subset of bit indices of a given range leads to consideration of a smaller set of values and potentially more efficient representations in ternary entries than the originally given range. But this is not enough, for correctness, we need that a matched value in the original range continues to be matched the corresponding entity on the representing subset of bit indices. In this section we introduce such entities implementing both efficiency and correctness that we call *subranges* and study their properties.

Consider a non-empty subset of bit indices $B \subseteq \{1, 2 \ldots, w\}$ of a w-bit range I. For a value x, denote by x^B a value obtained from x by taking the values of bits at the positions in B. Also we denote by I^B a set of values on B bit indices obtained from the values in a range I.

Example 3.1. Consider a 5-bit range $I = [23, 25]$. The following table consists of values $x \in I$ and $x^B \in I^B$ for a subset of bit-indices $B = \{1, 3, 5\}$; $I^B = \{4, 5, 7\}$.

x	23	24	25
x^B	7 (111_2)	4 (100_2)	5 (101_2)

In the following we explore structural properties of I^B in details. First, note that I^B is not necessarily a range, in Example 3.1, $I^B = \{4, 5, 7\}$. Second, for a range containing a single value $I = [l, l]$, the corresponding $I^B = [l^B, l^B]$ contains also a single value. Next we explore I^B properties on ranges containing at least two values.

For a range $I = [l, r]$, let $sim(I)$ be a first bit index whose value differs in the binary representations of l and r; in Example 3.1 $sim(I) = 2$. We say that a w-bit range $I = [l, r]$ is a *left border range* if l **mod** $2^{w-sim(I)} = 0$; and I is a *right border range* if $(r + 1)$ **mod** $2^{w-sim(I)} = 0$. For instance, $I = [24(11000_2), 28(11100_2)]$ is a left border range; and $I = [19(10011_2), 23(10111_2)]$ is a right border range.

Observation 1. *A range $I = [l, r]$ can be represented as a union of a right border range $I_1 = [l, l$ **or** $(2^{w-sim(I)} - 1)]$[1] and a left border range $I_2 = [r - r$ **mod** $2^{w-sim(I)}, r]$.*

For instance, $[9, 14]$ is a union of $I_1 = [9, 11]$ and $I_2 = [12, 14]$. To understand the properties of I^B, we are starting with a special case of a range I when I is a left (right) border range.

Lemma 3.2. *For a subset of bit-indices $B \subseteq \{1, \ldots, w\}$ and a w-bit range $I = [l, r]$, if I is a left (right) border range, the corresponding I^B is a $|B|$-bit left (right) border range.*

Proof. We show the lemma only for the case when I is a left border range. For the right border range the proof is symmetric. Note that it is sufficient to show the lemma only for the case when $sim(I) = 1$ (i.e., $l = 0$) since in other cases the bits at the first $sim(I) - 1$ positions coincide for all values belonging to I. For a range $I = [0, r]$, consider two values x, y such that $x < r$ and $y = x + 1$; then, either $y^B = x^B + 1$ or $y^B \leq x^B$. Thus, I^B is a range $[0, m^B]$, where m^B is a maximal value belonging to I^B. □

Now we are ready to show that the structure of I^B is similar to the structure of a regular range I; namely, both I^B and I can be represented as a union of the left and right border ranges despite the fact that I^B is not necessary a range.

[1] **or** is a bitwise operation

Algorithm 1 Construction of I^B for w-bit range $I = [l, r]$

```
1:  procedure IS_DIFFERENT_BIT(l, r, w, x)
2:      if (l and 2^(w−x)) ≠ (r and 2^(w−x)) then
3:          return true
4:      else
5:          return false
6:  procedure REDUCE_LEFT(I=[l, r], B)
7:      p = {x : x ∉ B, IS_DIFFERENT_BIT(l, r, w, x)}
8:      if p = ∅ then return [l^B, r^B]
9:      m = r or (2^(w−min(p)) − 1) − 2^(w−min(p))
            ▷ set (w-min(p))th bit of r to 0 and bits after (w − min(p))th to 1
10:     return [l^B, m^B]
11: procedure REDUCE_RIGHT(I=[l, r], B)
12:     p = {x : x ∉ B, IS_DIFFERENT_BIT(l, r, w, x)}
13:     if p = ∅ then return [l^B, r^B]
14:     m = l − l mod (2^(w−min(p))) + 2^(w−min(p))
            ▷ set (w-min(p))th bit of l to 1 and bits after (w − min(p))th to 0
15:     return [m^B, r^B]
16: procedure REDUCE(I=[l, r], B)
17:     if l = r then return [l^B, r^B]
18:     I_1 = [l, l or (2^(w−sim(I)) − 1)]
19:     I_2 = [r − r mod 2^(w−sim(I)), r]
                                  ▷ Split I into two border ranges I_1 and I_2
20:     return REDUCE_RIGHT(I_1, B) ∪ REDUCE_LEFT(I_2, B)
```

Theorem 3.3. *For a subset of bit-indices $B \subseteq \{1, \ldots, w\}$ and a w-bit range I, the set I^B of values can be represented as a union of the $|B|$-bit left border range and $|B|$-bit right border range.*

Proof. By Observation 1 a range I can be splitted into the left and right border ranges. Applying Lemma 3.2 for these border ranges, the theorem immediately follows. □

By definition we knew that in general I^B is a set of values, Theorem 3.3 sheds a light on internals of I^B. From now on, we call a set I^B of values as a *sub-range* of a w-bit range I on bit-indices $B \subseteq \{1, \ldots, w\}$. Algorithm 1 shows how to construct I^B for a given range I and a subset of bit-indices B.

Lemma 3.4. *For a set of indices $B \subseteq \{1, \ldots, w\}$ and a left (right) border range I, the procedure REDUCE_LEFT (REDUCE_RIGHT) in Algorithm 1 correctly constructs I^B.*

Proof. We show the lemma only for the case when I is a left border range; for the right border range the proof is symmetric. Since I is a left border range, $I^B = [l^B, m^B]$ for some $m \in I$. Denote by t a minimal bit index such that $t \notin B$ and the binary representations of l and r differ at the position t. Since I is a left border range values of bit of l and r at the position t equals 0 and 1, respectively. For r and m, the values of bits at the bit positions in B preceding t coincide; otherwise, there exists a bit index $t' < t$ such that $t' \notin B, l$ **and** $2^{w-t'} \neq r$ **and** $2^{w-t'}$ contradicting to minimality of t. Since a bit at the tth position is not considered during construction of I^B, we can obtain m from r by setting in a binary representation of r the bit value at the tth position to

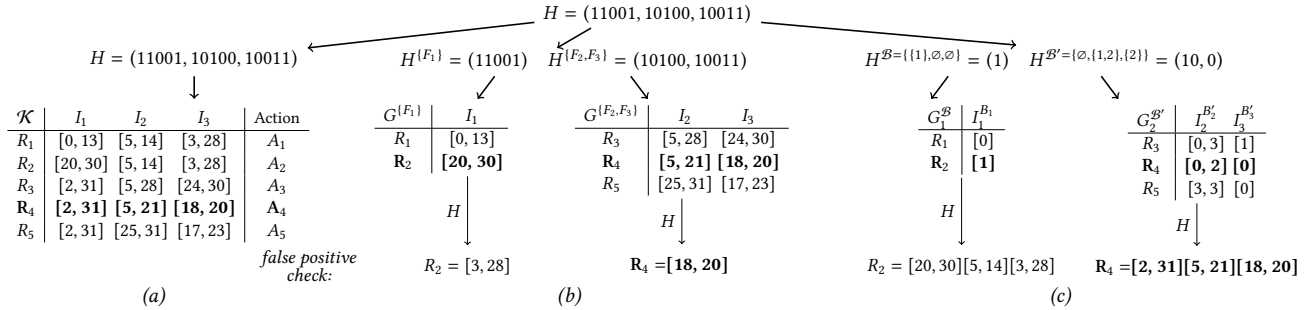


Figure 2: Multigroup representations our RR method vs. SAX-PAC [10]: (a) an original classifier $\mathcal{K}$ with 3-field rules, each field is on 5 bits; (b) SAX-PAC representation of $\mathcal{K}$ on two groups: the first group rules are disjoint on a single field; the second group rules are disjoint on two fields; (c) the proposed RR implementing rule disjointness on a single bit in the first group and on three bits in the second group.

0 and the bits at the positions succeding t to 1. If there is no such position t then $m = r$. □

Theorem 3.5. *For a given subset of indices $B \subseteq \{1, \ldots, w\}$ and a w-bit range $I = [l, r]$, the procedure* REDUCE *in Algorithm 1 correctly calculates I^B in $O(w)$ time.*

Proof. If a range I contains a single value, $I^B = [l^B, l^B]$, otherwise I can be splitted into the left and right border ranges I_1, I_2 and then their sub-ranges are constructed correctly by Lemma 3.4. The running time immediately follows by construction of Algorithm 1 □

Recall that encoding of a w-bit regular range consists of $2 \cdot w - 2$ ternary bit strings for the prefix expansion and $2 \cdot w - 4$ for SRGE in the worst case. The following theorem shows that despite the fact that a subrange I^B is not necessary a range, its ternary encoding consists of at most $2 \cdot |B| - 2$ ternary bit strings for both prefix expansion and SRGE encoding.

Theorem 3.6. *A subrange I^B can be encoded by at most $2 \cdot |B| - 2$ ternary bit strings using prefix expansion or SRGE encoding.*

Proof. The correctness of the theorem immediately follows from the fact that prefix expansion or SRGE encoding of a single w-bit left (right) border range consists of at most $w - 1$ ternary bit strings. □

Note that we can operate on subranges as on regular ranges since the intersection of two subranges can be verified in a constant time. Now we are ready to move on to equivalent representations of multi-field range-based classifiers.

4 EQUIVALENT REPRESENTATIONS

In difference from the previous works considering equivalent classifiers by encoding all ranges of every field [3, 17] (to list a few), [10] proposes equivalent multigroup representations

	range encoding methods			
	prefix [17]		SRGE [3]	
	# entries	# bits	# entries	# bits
$\mathcal{K}$	358	5370	188	2820
$G^{\{F_1\}} + G^{\{F_2,F_3\}}$ [10]	44	405	35	320
$G_1^{\mathcal{B}} + G_2^{\mathcal{B}'}$, RR	6	14	6	14

Figure 3: Memory requirements for the reprsentations in Fig. 2.

(not classifiers). These representations consist of at most β lookup tables implementing rule disjointness on a subset of fields followed by false-positive checks, where β is a constant corresponding to a number of "pseudo-parallel" lookups that can be issued at line-rate. Representations in [10] allow to encode only a subset of ranges participating in lookup tables in difference from equivalent classifiers requiring encoding all range fields of every rule. Because of rule disjointness only a single rule can be matched at each lookup table; hence, for an incoming header, the false-positive check is done only on matched rules whose ranges can be verified without encoding. Multiple lookup tables (a *multi-group representation*) are necessary to deal with general classifiers whose rules can intersect on all fields and to improve representation efficiency. Note that different lookup tables (groups) can use different subsets of fields to implement rule disjointness.

Formally, consider a classifier $\mathcal{K}$ with k fields represented by ranges. Let $\mathcal{G}$ be a set of β disjoint groups containing rules from $\mathcal{K}$, where every group implements rule disjointness on at most $k-1$ fields. Since not all rules can be covered by $\mathcal{G}$ for a given classifier $\mathcal{K}$, there is a portion of remaining rules C from $\mathcal{K}$ not belonging to $\mathcal{G}$. The classification process in [10] is the following: (1) find a classifying rule at every group (lookup table) and perform a false-positive check for every matched rule (at most β overall); (2) independently with (1) find a classifying rule in C; (3) from at most β matched rules passing a false-positive check and a classifying rule in C return the action of a rule with the highest priority or the default action if there is no a single matching rule for a given header. The classification process is depicted in

Algorithm 2 Heuristic for SAX-PAC [10] representation

```
1: procedure GROUP_BUILD(𝒦, ℒ)
2:     G = {}
3:     for R ∈ 𝒦 do
4:         if R^ℒ is disjoint to all R'^ℒ : R' ∈ G then
5:             G = G ∪ {R}
6:     return G
7: procedure FIND_MAX_GROUP(𝒦, l)
8:     ℒ = a set of l fields maximizing GROUP_BUILD(𝒦, ℒ)
9:     return (GROUP_BUILD(𝒦, ℒ), ℒ)
10: procedure GREEDY_GROUP(𝒦, l)
11:     𝒢 = {}
12:     while 𝒦 ≠ ∅ do
13:         G, ℒ = FIND_MAX_GROUP(𝒦, l)
14:         𝒦 = 𝒦 \ G;
15:         𝒢 = 𝒢 ∪ {G^ℒ}
16:     return 𝒢
```

Fig. 2b. In [10] the authors show that this representation is equivalent to an originally given k-field classifier whose fields are represented by ranges.

Since our goal is to demonstrate advantages of per-bit over per-field resolution, we follow the same classification process and consider the similar representation $(\mathcal{G}, C)$ but now each group implements rule disjointness on a subset of subranges (at most one per field) on $\mathcal{B} = \{B_1, B_2, \ldots, B_k\}$, where B_i is a subset of bit indices of a field F_i. Note that B_i can be empty meaning that the corresponding field does not participate in the lookup table. Observe that different groups in $\mathcal{G}$ can implement rule disjointness on different $\mathcal{B}$s. The classification process is illustrated in Fig. 2c. Similarly to the representation with per-field resolution this representation is equivalent to an originally given classifier with two differences: the matching property of subranges and the false-positive check that should be done on all k fields (and not only on the remaining fields as in [10]). The impact of per-bit versus per-field resolution can be seen in Fig. 3.

We assume that both $\mathcal{G}$ and C (if C exists) are implemented in TCAM. For a given range encoding method E and a classifier $\mathcal{K}^{\mathcal{B}}$ on $\mathcal{B}$ bit indices, denote by $E(\mathcal{K}^{\mathcal{B}})$ the total size in bits of all ternary rules constructed from $\mathcal{K}^{\mathcal{B}}$. Similarly, the size $E(\mathcal{G})$ is a total size of containing groups.

Problem 1. *For a given range encoding method E and a range-based classifier $\mathcal{K}$ find a multigroup representation $(\mathcal{G}, C)$ minimizing $E(\mathcal{G}) + E(C)$.*

Even in the special case of a $\beta = 1$ this problem is intractable that can be shown by reduction from SetCover [8].

5 EVALUATION STUDY

In this section we explore the impact of subranges on efficiency of classifier representations.

Evaluated heuristics. Since we want to demonstrate the impact of range representations with per-bit versus per-field resolution, we use the same heuristic as in SAX-PAC [10] to build a multi-group representation; we consider two cases:

Algorithm 3 Heuristic for RR representation

```
1: procedure REDUCE_GROUP(G)
2:     ℬ = set of sets of all bit-indices of all fields of G
3:     while it is possible do
4:         remove a bit-index from one of sets in ℬ minimizing E(G^ℬ)
           and preserving pairwise disjointness of all rules in G^ℬ.
5:     return G^ℬ
6: procedure GREEDY_GROUP_BIT(𝒦, l)
7:     𝒢 = {}
8:     while 𝒦 ≠ ∅ do
9:         G, ℱ = FIND_MAX_GROUP(𝒦, l)          ▷ ℱ is ignored
10:        𝒦 = 𝒦 \ G;
11:        𝒢 = 𝒢 ∪ {REDUCE_GROUP(G)}
12:    return 𝒢
```

at every group the rules are pairwise disjoint based on one or two fields ($l = 1$ or $l = 2$); see GREEDY_GROUP$(\mathcal{K}, l)$ in Algorithm 2. Since SAX-PAC implements per-field resolution for range-based fields, to be stored in TCAM, they are encoded into ternary entries by one of the encoding methods (in our case prefix [17] or SRGE [3]). The proposed RR starts with the same assignment of $\mathcal{K}$ rules into multiple groups as in SAX-PAC but now for every range-based field a subrange is found minimizing the total size in bits or in ternary entries; see GREEDY_GROUP_BIT$(\mathcal{K}, l)$ in Algorithm 3. For cases when $l \leq 2$ the running time of Algorithm 2 is $O(N^2 \cdot k^2 \cdot \beta)$ and the running time of Algorithm 3 is $O(N^2 \cdot k^2 \cdot \beta + N^2 \cdot w^2)$, where N is a number of rules in $\mathcal{K}$.

Methodology. Since the goal of this paper is to understand design principals to represent desired levels of expressiveness and scalability, we assume the extreme case when all classifier fields are represented by ranges. Each synthetically generated classifier consists of 10000 rules that are generated independently. Each field range $I = [l, r]$ is generated independently according to the following distribution: (1) the value of $sim(I)$ is chosen uniformly at random; (2) the range bounds l and r are generated uniformly such that the binary representations of l and r coincide on the first $sim(I) - 1$ bits. Though other distributions are possible, we choose this one to simulate sparseness of covered range values. In our experiments we vary a number of range-based fields in a rule and a range width in bits. For both SRGE and prefix range encodings, we compare the total size in bits and entries among (1) conventional equivalent ternary classifiers, (2) multi-group representations with per-field resolution as in SAX-PAC (the columns SAX-PAC in Fig. 4 and Fig. 5), and (3) multigroup representations with per-bit resolution constructed by GREEDY_GROUP_BIT (the columns RR in Fig. 4 and Fig. 5). We release the code for our evaluation study as an open source [7].

Impact of range width. In the following denote by $\mathcal{G}_1$ and $\mathcal{G}_2$ multi-group representations implementing rule disjointness on one and two fields, respectively. For classifier rules based on four 16-bit ranges encoded by prefix expansion, the total size of $\mathcal{G}_1$ with per-field resolution is 1.73 times

Input classifier $\mathcal{K}$				Multigroup: rule disjointness on 1 field						Multigroup: rule disjointness on 2 fields					
characteristics		ternary, prefix		# groups		SAX-PAC [10]		RR		# groups		SAX-PAC [10]		RR	
# ranges	range width	# entries	# bits	95% of $\mathcal{K}$	100% of $\mathcal{K}$	# entries	# bits	# entries	# bits	95% of $\mathcal{K}$	100% of $\mathcal{K}$	# entries	# bits	# entries	# bits
4	16	$3.4 \cdot 10^7$	$2.2 \cdot 10^9$	24	44	40528	648448	23310	505876	7	18	370852	$1.2 \cdot 10^7$	128712	$4.1 \cdot 10^6$
4	20	$8.4 \cdot 10^7$	$6.7 \cdot 10^9$	11	26	54375	$1.1 \cdot 10^6$	18157	417708	5	14	632983	$2.5 \cdot 10^7$	68043	$2.1 \cdot 10^6$
4	24	$1.8 \cdot 10^8$	$1.7 \cdot 10^{10}$	8	20	71220	$1.7 \cdot 10^6$	15286	363079	4	11	983914	$4.7 \cdot 10^7$	52304	$1.6 \cdot 10^6$
4	28	$3.4 \cdot 10^8$	$3.9 \cdot 10^{10}$	6	17	87787	$2.5 \cdot 10^6$	14177	328468	3	10	$1.4 \cdot 10^6$	$7.9 \cdot 10^7$	44735	$1.7 \cdot 10^6$
4	32	$5.9 \cdot 10^8$	$7.5 \cdot 10^{10}$	4	13	106248	$3.4 \cdot 10^6$	12688	317128	3	9	$1.9 \cdot 10^6$	$1.2 \cdot 10^8$	29981	$1.0 \cdot 10^6$
3	32	$3.8 \cdot 10^7$	$3.7 \cdot 10^9$	5	18	109115	$3.5 \cdot 10^6$	18121	451028	3	11	$2.1 \cdot 10^6$	$1.3 \cdot 10^8$	71387	$2.3 \cdot 10^6$
4	32	$5.8 \cdot 10^8$	$7.4 \cdot 10^{10}$	4	11	106315	$3.4 \cdot 10^6$	13998	371824	2	7	$1.9 \cdot 10^6$	$1.2 \cdot 10^8$	31027	$1.0 \cdot 10^6$
5	32	$9.1 \cdot 10^9$	$1.5 \cdot 10^{12}$	4	9	104343	$3.3 \cdot 10^6$	11521	301740	2	7	$1.9 \cdot 10^6$	$1.2 \cdot 10^8$	15523	464471
6	32	$1.4 \cdot 10^{11}$	$2.7 \cdot 10^{13}$	5	9	105026	$3.4 \cdot 10^6$	11160	288901	3	5	$1.9 \cdot 10^6$	$1.2 \cdot 10^8$	13882	464171

Figure 4: Total size of ternary rules in entries and in bits by prefix expansion.

Input classifier $\mathcal{K}$				Multigroup: rule disjointness on 1 field						Multigroup: rule disjointness on 2 fields					
characteristics		ternary, SRGE		# groups		SAX-PAC [10]		RR		# groups		SAX-PAC [10]		RR	
# ranges	range width	# entries	# bits	95% of $\mathcal{K}$	100% of $\mathcal{K}$	# entries	# bits	# entries	# bits	95% of $\mathcal{K}$	100% of $\mathcal{K}$	# entries	# bits	# entries	# bits
4	16	$2.1 \cdot 10^7$	$1.4 \cdot 10^9$	24	44	34620	553920	19053	405194	7	18	286856	$9.2 \cdot 10^6$	87677	$2.8 \cdot 10^6$
4	20	$5.8 \cdot 10^7$	$4.6 \cdot 10^9$	11	26	47328	946560	16084	374638	5	14	512044	$2.0 \cdot 10^7$	46351	$1.5 \cdot 10^6$
4	24	$1.3 \cdot 10^8$	$1.3 \cdot 10^{10}$	8	20	63513	$1.5 \cdot 10^6$	14055	341758	4	11	824121	$4.0 \cdot 10^7$	37976	$1.2 \cdot 10^6$
4	28	$2.6 \cdot 10^8$	$2.9 \cdot 10^{10}$	6	17	79694	$2.2 \cdot 10^6$	12903	305834	3	10	$1.2 \cdot 10^6$	$6.8 \cdot 10^7$	40733	$1.5 \cdot 10^6$
4	32	$4.6 \cdot 10^8$	$5.9 \cdot 10^{10}$	4	13	97917	$3.1 \cdot 10^6$	11899	292533	3	9	$1.7 \cdot 10^6$	$1.1 \cdot 10^8$	24192	775260
3	32	$3.2 \cdot 10^7$	$3.1 \cdot 10^9$	5	18	100641	$3.2 \cdot 10^6$	16286	399179	3	11	$1.8 \cdot 10^6$	$1.2 \cdot 10^8$	58765	$1.9 \cdot 10^6$
4	32	$4.5 \cdot 10^8$	$5.8 \cdot 10^{10}$	4	11	97835	$3.1 \cdot 10^6$	13263	350490	2	7	$1.7 \cdot 10^6$	$1.1 \cdot 10^8$	24976	805932
5	32	$6.7 \cdot 10^9$	$1.1 \cdot 10^{12}$	4	9	95834	$3.1 \cdot 10^6$	11229	285668	2	7	$1.7 \cdot 10^6$	$1.1 \cdot 10^8$	14516	444518
6	32	$1.0 \cdot 10^{11}$	$1.9 \cdot 10^{13}$	5	9	96610	$3.1 \cdot 10^6$	10830	284829	3	5	$1.6 \cdot 10^6$	$1.0 \cdot 10^8$	12355	380373

Figure 5: Total size of ternary rules in entries and in bits by SRGE.

bigger than with per-bit; this ratio is rapidly growing when range width is increasing; e.g., for 32-bit ranges the same ratio is already equal to 10. The similar effect is seen on the total size in bits for the both range encoding methods. For $\mathcal{G}_2$ representations, the effect of per-bit resolution is even more pronounced; e.g., for classifier rules consisting of four 32-bits ranges encoded by SRGE, the ratio between per-field and per-bit resolution in ternary entries is already 70 times since the total size of $\mathcal{G}_2$ depends quadratically on range width.

Number of fields in a rule. Increasing the number of fields has no significant effect on the total size of multi-group representations with per-field resolution and this is a significant advantage of SAX-PAC versus equivalent classifiers encoding all ranges. The per-bit resolution provides additional memory saving since it allows to pickup multiple subranges minimizing the total size; e.g., for classifier rules on six 32-bit ranges, the size of SRGE encoding for $\mathcal{G}_1$ with per-bit resolution is 10830 ternary entries which is very close to the optimal case when every range-based rule is encoded by a single ternary entry; from the other hand, $\mathcal{G}_1$ with per-field resolution and SRGE encoding requires 96610 entries. In the worst case among all experiments, the average number of ternary entries in $\mathcal{G}_1$ with per-bit resolution does not exceed 2.4 entries per rule.

Number of groups. $\mathcal{G}_2$ requires smaller number of groups than $\mathcal{G}_1$ when per-field resolution is considered; from the other hand, the total size of encoded ternary entries is significantly bigger for $\mathcal{G}_2$ than for $\mathcal{G}_1$; e.g., for rules on five 32-bit ranges, the total size of $\mathcal{G}_2$ in entries for SRGE encdoing is 17 times bigger than for $\mathcal{G}_1$. For per-bit resolution, this effect is less pronounced and does not exceed 6 times in all experiments. Note that in all experiments for classifiers with at least 24-bits range width, the number of entries required for $\mathcal{G}_2$ with per-bit resolution is smaller even than for $\mathcal{G}_1$ with per-field resolution showing effectiveness of per-bit resolution for reduction not only of the total size (both in entries and bits) but also of the number of required groups.

6 CONCLUSION

Range-based field representation is an important abstraction to balance between scalability and expressiveness in packet classifiers; when a number of range-based fields is growing, equivalent classifiers encoding all ranges is not the right direction. Equivalent representations (not classifiers) as SAX-PAC [10] significantly improve memory requirements but still operating with per-field resolution can demand significant memory resources and the number of groups to implement desired structural properties. In this paper we introduce a notion of subranges allowing to operate on ranges with per-bit resolution and overcome constraints of per-field representations. Coexistence of subranges with other encoding methods as [1, 2, 5] and software-based classifier representations in regular memory we leave for the future study.

Acknowledgements. This project has been made possible in part by a grant from the Cisco University Research Program Fund, an advised fund of Silicon Valley Community Foundation. We thank the anonymous reviewers and our shepherd Ran Ben-Basat for their insightful comments.

REFERENCES

[1] Anat Bremler-Barr, Yotam Harchol, David Hay, and Yacov Hel-Or. 2018. Encoding Short Ranges in TCAM Without Expansion: Efficient Algorithm and Applications. *IEEE/ACM Trans. Netw.* 26, 2 (2018), 835–850.

[2] Anat Bremler-Barr, David Hay, and Danny Hendler. 2012. Layered interval codes for TCAM-based classification. *Computer Networks* 56, 13 (2012), 3023–3039.

[3] Anat Bremler-Barr and Danny Hendler. 2012. Space-Efficient TCAM-Based Classification Using Gray Coding. *Trans. Computers* 61, 1 (2012), 18–30.

[4] Yeim-Kuan Chang and Cheng-Chien Su. 2007. Efficient TCAM Encoding Schemes for Packet Classification Using Gray Code. In *GLOBECOM*. 1834–1839.

[5] Hao Che, Zhijun Wang, Kai Zheng, and Bin Liu. 2008. DRES: Dynamic Range Encoding Scheme for TCAM Coprocessors. *Trans. Computers* 57, 7 (2008), 902–915.

[6] Pavel Chuprikov, Kirill Kogan, and Sergey I. Nikolenko. 2017. General ternary bit strings on commodity longest-prefix-match infrastructures. In *ICNP*. 1–10.

[7] How to deal with range-based packet classifiers [n. d.]. How to deal with range-based packet classifiers. https://github.com/sosrranges/subranges.

[8] Richard M. Karp. 1972. *Reducibility among Combinatorial Problems*. Springer US, Boston, MA, 85–103.

[9] Kirill Kogan, Sergey I. Nikolenko, Patrick Eugster, Alexander Shalimov, and Ori Rottenstreich. 2016. FIB efficiency in distributed platforms. In *ICNP*. 1–10.

[10] Kirill Kogan, Sergey I. Nikolenko, Ori Rottenstreich, William Culhane, and Patrick Th. Eugster. 2016. Exploiting Order Independence for Scalable and Expressive Packet Classification. *Trans. Networking* 24, 2 (2016), 1251–1264.

[11] Karthik Lakshminarayanan, Anand Rangarajan, and Srinivasan Venkatachary. 2005. Algorithms for advanced packet classification with ternary CAMs. In *SIGCOMM*. 193–204.

[12] Netlogic Microsystems. Content addressable memory [n. d.]. Netlogic Microsystems. Content addressable memory. http://www.netlogicmicro.com.

[13] Sergey I. Nikolenko, Kirill Kogan, Gábor Rétvári, Erika R. Bérczi-Kovács, and Alexander Shalimov. 2016. How to represent IPv6 forwarding tables on IPv4 or MPLS dataplanes. In *INFOCOM Workshops*. 521–526.

[14] OpenFlow 1.3 specification, 2012 2012. OpenFlow 1.3 specification, 2012. http://www.openflow.org/wk/index.php/OpenFlow_1_3_proposal.

[15] $P4_{16}$ Language Specification 2017. $P4_{16}$ Language Specification. https://p4.org/p4-spec/docs/P4-16-v1.0.0-spec.pdf.

[16] Ori Rottenstreich, Rami Cohen, Danny Raz, and Isaac Keslassy. 2013. Exact Worst Case TCAM Rule Expansion. *IEEE Trans. Computers* 62, 6 (2013), 1127–1140.

[17] Venkatachary Srinivasan, George Varghese, Subhash Suri, and Marcel Waldvogel. 1998. Fast and Scalable Layer Four Switching. In *SIGCOMM*. 191–202.