Optimal In/Out TCAM Encodings of Ranges

Ori Rottenstreich, Isaac Keslassy, *Senior Member, IEEE*, Avinatan Hassidim, Haim Kaplan, and Ely Porat

***Abstract*—Hardware-based packet classification has become an essential component in many networking devices. It often relies on ternary content-addressable memories (TCAMs), which compare the packet header against a set of rules. TCAMs are not well suited to encode range rules. Range rules are often encoded by multiple TCAM entries, and little is known about the smallest number of entries that one needs for a specific range. In this paper, we introduce the *In/Out TCAM*, a new architecture that combines a regular TCAM together with a modified TCAM. This custom architecture enables independent encoding of each rule in a set of rules. We provide the following theoretical results for the new architecture: 1) We give an upper bound on the worst-case expansion of range rules in one and two dimensions. 2) For extremal ranges, which are 89% of the ranges that occur in practice, we provide an efficient algorithm that computes an optimal encoding. 3) We present a closed-form formula for the average expansion of an extremal range.**

***Index Terms*—Optimal range encoding, packet classification, ternary content-addressable memory (TCAM).**

I. INTRODUCTION

A. Background

Packet classification is the key function behind many network applications, such as routing, filtering, security, accounting, monitoring, load balancing, policy enforcement, differentiated services, virtual routers, and virtual private networks [1]–[4]. For each incoming packet, a packet classifier compares the packet header fields against a list of rules, e.g., from access control lists (ACLs), returns the first rule that matches the header fields, and applies a corresponding action to the packet. Typically, a tuple of five fields from the packet header is used, namely the source IP address, destination IP address, source port number, destination port number, and protocol type.

Today, hardware-based *ternary content-addressable associative memories* (TCAMs) are the standard devices for high-speed packet classification [5], [6]. They match the concatenation of the five-tuple from the packet header into a fixed-width ternary array composed of 0's, 1's, and $*$'s (don't-care bits). For each packet, a TCAM device checks all the rules in parallel, and therefore reaches higher rates than other software-based or hardware-based classification algorithms [1]–[3], [7], [8].

There are two types of rules: simple rules that specify a fixed value (or a specific prefix range as defined formally below) for each field of the header, and *range rules*. Typically, a *range rule* applies when the source port and/or the destination port need to be in specific intervals. TCAMs are not well suited for the representation of range rules. Encoding range rules often requires several TCAM entries (the number of entries is called the range expansion), and therefore although most rules are simple rules, most TCAM entries are used to encode range rules [9]. In addition, there is an evidence that the percentage of range rules is increasing [10].

As most range rules cannot be encoded with a single TCAM entry, the common approach to encode range rules in a TCAM is by a set of disjoint prefix ranges. This requirement of using several TCAM entries becomes more critical for rules with ranges defined over two fields. In some cases, this simple encoding requires $(2 \cdot 16 - 2)^2 = 900$ entries to encode two 16-bit ranges restricting the values of the source port and the destination port [11]. In contrast, advanced coding techniques that can encode a range by first eliminating its complement have worst-case expansion that is linear in the number of bits required to specify the range [12]. However, a conventional TCAM does not allow to encode a classifier with several rules by simply concatenating their advanced encodings.

Nevertheless, understanding how to encode a single rule with a small expansion is a fundamental question that was studied intensively. To date, *there is no polynomial-time algorithm that computes an optimal encoding (encoding that minimizes the number of TCAM entries) for any range rule*. Instead, past work suggested to compute restricted encoding: either an encoding using only prefix ranges [13], [14] or an encoding that encodes the range itself and not its complement [15] (i.e., using only *in* entries in the terminology below). Other papers used heuristic approaches [1]–[3], [16]–[24].

B. Our Contributions

In this paper, we study the fundamental complexity of encoding a range rule, using entries each marked *in* or *out*. A header is in the range if and only if it matches at least one entry and the first entry it matches is an *in* entry. We give lower and upper bounds on the size of the optimal encoding of any range,

Manuscript received March 17, 2013; revised October 30, 2013;June 25, 2014;and November 10, 2014; accepted November 11, 2014; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor A. X. Liu. Date of publication January 09, 2015; date of current version February 12, 2016. This work was supported in part by the Israel Science Foundation under Grants No. 822/10 and No. 1241/12, the United States-Israel Binational Science Foundation under Project No. 2012344, the German-Israeli Foundation for Scientific Research and Development (GIF) under Grant No. 1161/2011, the Israeli Centers of Research Excellence (I-CORE) (center No. 4/11), the Gordon Fund for Systems Engineering, the Neptune Magnet Consortium, the Israel Ministry of Science and Technology, the Intel ICRI-CI Center, the Hasso Plattner Institute Research School, the Technion Funds for Security Research, and the Erteschik and Greenberg Research Funds.

O. Rottenstreich is with Princeton University, Princeton, NJ 08540 USA (e-mail: orir@cs.princeton.edu).

I. Keslassy is with the Technion, Haifa 32000, Israel (e-mail: isaac@ee.technion.ac.il).

A. Hassidim and E. Porat are with Bar-Ilan University, Ramat Gan 52900, Israel (e-mail: avinatan@cs.biu.ac.il; porately@cs.biu.ac.il).

H. Kaplan is with Tel Aviv University, Tel Aviv 69978, Israel (e-mail: haimk@post.tau.ac.il).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2014.2382031

TABLE I
SUMMARY OF OUR NEW RESULTS (IN BOXED BOLD) IN COMPARISON TO PREVIOUS RESULTS: (a) (LAST ROW) WE GIVE AN OPTIMAL ALGORITHM FOR THE ENCODING OF ONE-DIMENSIONAL (GENERALIZED) EXTREMAL RULES (RANGES OF THE FORM $[0, y]$). THIS IS THE FIRST OPTIMAL RESULT WHEN THE DEGREES OF FREEDOM OF THE ALGORITHM ARE NOT LIMITED. (b) (LAST ROW) WE GIVE A TIGHT BOUND OF W ON THE EXPANSION OF GENERAL ONE-DIMENSIONAL RULES WHEN THE DEGREES OF FREEDOM OF THE ALGORITHM ARE NOT LIMITED. IN ADDITION, FOR TWO-DIMENSIONAL RULES, WE GIVE A TIGHT BOUND OF $W + 1$ FOR EXTREMAL RULES (ASSUMING EVEN W FOR SIMPLICITY), AND A TIGHT BOUND OF $2W$ FOR GENERAL RULES

(a) Optimal algorithm for any range

Constraints			References	Extremal Ranges		General Ranges	
No *out* entries	Prefix code	Gray code		One Dimension	Two Dimensions	One Dimension	Two Dimensions
x	x	-	[11]	$\surd$	-	$\surd$	-
x	-	x		-	-	-	-
x	-	-		-	-	-	-
-	x	-	[13], [14]	$\surd$	$\surd$	$\surd$	$\surd$
-	-	-		$\boxed{\surd}$	-	-	-

(b) Bounds on worst-case expansion over all ranges

Constraints			References	Extremal Ranges				General Ranges			
No *out* entries	Prefix code	Gray code		One Dimension		Two Dimensions		One Dimension		Two Dimensions	
				Upper Bound	Lower Bound	Upper Bound	Lower Bound	Upper Bound	Lower Bound	Upper Bound	Lower Bound
x	x	-	[11]	W	W	W^2	-	$2W-2$	$2W-2$	$(2W-2)^2$	-
x	-	x	[9]	-	-	-	-	$2W-4$	W	$(2W-4)^2$	-
x	-	-	[15]	W	-	W^2	-	$2W-4$	$2W-4$	$(2W-4)^2$	-
-	x	-	[12], [13]	$\left\lceil\frac{W+1}{2}\right\rceil$	$\left\lceil\frac{W+1}{2}\right\rceil$	$\boxed{\mathbf{W+1}}$	$\boxed{\mathbf{W+1}}$	W	W	$\boxed{\mathbf{2W}}$	$\boxed{\mathbf{2W}}$
-	-	-	[12]	$\left\lceil\frac{W+1}{2}\right\rceil$	$\left\lceil\frac{W+1}{2}\right\rceil$	$\boxed{\mathbf{W+1}}$	$\boxed{\mathbf{W+1}}$	W	$\boxed{\mathbf{W}}$	$\boxed{\mathbf{2W}}$	$\boxed{\mathbf{2W}}$

and we also develop practical algorithms for encoding a given range with a guarantee on its expansion.

Our first contribution in this paper is a new *In/Out TCAM architecture*. The new architecture combines a regular TCAM and a modified TCAM. Simple rules are encoded in the regular TCAM using only *in* entries. Complex rules are encoded in the modified TCAM, using both *in* and *out* entries. The In/Out TCAM architecture allows independent encoding of each rule without considering their mutual interactions. This architecture reduces the expansion of complex rules, albeit with an added implementation cost, since the In/Out TCAM architecture relies on additional logic and *is not a simple off-the-shelf TCAM*.

Our second contribution is a theoretical result motivated by the In/Out TCAM architecture. Consider a set $\{0, 1, \cdots, 2^W - 1\}$ of 2^W points, also represented as a binary tree with 2^W leaves. *Extremal ranges* are ranges that cover the first or last leaves of the tree, i.e., ranges of the form $[0, y]$ or $[y, 2^W - 1]$. *Generalized extremal ranges* are ranges that are extremal for the minimal subtree containing their endpoints in this tree. (We later provide a more formal definition.) For instance, within a tree with 64 leaves ($W = 6$), [5, 7] is extremal for the subtree that represents [4, 7], and therefore is a generalized extremal range. Our second contribution is a simple linear-time algorithm that finds an optimal encoding for any given generalized extremal range. The main insight that allows us to obtain this result is a proof that there is an optimal TCAM encoding for generalized extremal ranges that uses only TCAM entries, each representing a prefix range. We can then use a simple dynamic programming algorithm to find the smallest TCAM encoding that uses only TCAM entries that represent prefix ranges, as the algorithm of [13].

This result is particularly appealing because the set of generalized extremal ranges is significant in practice. To estimate the potential impact of our results, we considered a union of 120 real-life classification databases from [1], [9], and [10], containing 214 941 rules. We find that *97.2% of these rules are generalized extremal rules* (i.e., 208 850 rules). Even after excluding the exact-match rules, which are trivial to encode, *89.4% of the non-exact-match rules are generalized extremal range rules* (51 065 rules out of 57 146).

Our discovery of an optimal algorithm for extremal ranges also allows us to analyze the *expected length of the optimal encoding* over all extremal ranges. We derive a closed-form formula for this expected expansion and show that asymptotically (when W is large) it is 2/3 of the worst case.

Our third contribution is that *we prove tight bounds on the worst-case expansion of any one-dimensional and two-dimensional range rule* [illustrated in the boxed bold values in the last two rows of Table I(b)]. Specifically, we give a simple algorithm that encodes any range by at most W entries, and we prove that there is a range that cannot be encoded by less than W entries. Our lower bound of W improves substantially on a best, previously known, lower bound of $\lceil (W+1)/2 \rceil$ and answers a question left open in [25]. We also present an algorithm that is optimal in the worst case for general two-dimensional range rules. Such rules include two ranges, one for the source port and the other for the destination port. If W is the length of each of the two fields, our algorithm encodes any two-dimensional range with at most $2W$ entries. We also prove that there exists a two-dimensional range whose encoding requires $2W$ TCAM entries.

As we mentioned, our study focuses on the encoding of a single range rule. Encodings of individual range rules with only *in* entries can be concatenated to get an encoding of a sequence of rules. This, unfortunately, is not true for encodings with both *in* and *out* entries. To combine these encodings of individual rules, we suggest the In/Out TCAM architecture described in Section III.

C. Related Work

As is further illustrated in Table I, several previous papers have tried to find bounds on the worst-case expansion of a single range rule. It is well known that each range defined over a field of W bits can be encoded by at most $2W - 2$ prefix TCAM entries (as defined formally below) for $W \geq 2$. In this encoding, all entries capture only inputs that are in the range (i.e., they are all *in* entries in our terminology) [11]. For example, assume that $W = 4$, and we want to encode the single range $R = [1, 14] \subseteq [0, 2^W - 1]$ so that packets in this range are matched by at least one entry while others are not matched. Then, we need $2W - 2 = 6$ TCAM entries, not counting the last default entry: $(0001 \rightarrow \text{in}, 001{*} \rightarrow \text{in}, 01{*}{*} \rightarrow \text{in}, 10{*}{*} \rightarrow \text{in}, 110{*} \rightarrow \text{in}, 1110 \rightarrow \text{in}, {*}{*}{*}{*} \rightarrow \text{out})$.

Using a non-prefix TCAM encoding, the upper bound of $2W - 2$ was improved to $2W - 4$ [26]. To show that, the disjunctive normal form representation of a range function was studied. Usually, a header value is represented by the binary code that keeps its base-2 value as a sequence of bits. Alternatively, the header value can be described in a Gray code in which any two adjacent values differ by a single bit. A Gray code can be used instead of a binary code to get the same improvement in the worst-case expansion from $2W - 2$ to $2W - 4$ [9].

The encoding that we mentioned above uses only *in* entries. When both *in* and *out* entries are allowed, the order of the entries is significant since the decision whether a packet is in the range is determined by the first matching entry. In [12] (a previous work written by some of the authors of the current paper), it is shown that when both *in* and *out* entries are allowed, the maximum expansion is W. For instance, the range $R = [1, 14]$ could be encoded using $3 \leq W$ entries: $(0000 \rightarrow \text{out}, 1111 \rightarrow \text{out}, {*}{*}{*}{*} \rightarrow \text{in})$.

Some rules specify a range both for the source IP and for the destination IP. This motivates considering rules that are the product of d ranges defined on d different fields of W bits each. It is easy to see that they can be simply encoded using up to $(2W - 2)^d$ prefix TCAM entries each matching some part of the range. This gives a bound of 900 TCAM entries for a pair of $(d = 2)$ port ranges of 16 bits each [1].

There are not many known lower bounds on the number of TCAM entries required to encode a range. If the encoding is restricted to use only *in* entries, then there is a range for which the encoding has to contain at least W entries [9]. Furthermore, for the binary code, it was shown in [15] that there is a range whose encoding requires at least $2W - 4$ *in* TCAM entries.

When both *in* and *out* entries are used, [12] presented a lower bound of $\lceil (W+1)/2 \rceil$ on the worst-case expansion of extremal ranges given in binary codes, even when the entries are not restricted to be prefix. For general ranges, a lower bound of W was given *only when the entries are restricted to be prefix* [12].

Algorithms for finding an optimal *prefix* encoding for a given range are presented in [13] and [14].

There is extensive literature on *efficient approaches* of how to encode ranges in TCAMs [1]–[3], [16]–[18]. In particular, several schemes such as [19]–[24], and [27]–[29] try to minimize the size of the encoding of a set of rules by exploiting the interactions between the rules. For instance, techniques to reduce the number of fields in a representation of a classifier were described in [29]. Besides TCAMs, coding schemes for the compression of forwarding tables have been described in [30]–[32].

II. Model and Notations

A. Terminology

We first formally define the terminology used in this paper. Unless mentioned otherwise, we assume a binary code representation. For simplicity, as long as there is no confusion, we also do not distinguish between a W-bit binary string (in $\{0,1\}^W$) and its value (in $[0, 2^W - 1]$). We denote by xy the concatenation of the strings x and y, and by $(x)^k$ the concatenation of k copies of the string x. We number the bits of a string from left to right, i.e., from the most significant to the least significant.

Definition 1 (Range, Prefix Range): A range R of width W is defined by two bit strings r_1 and r_2 of W bits each, such that $r_1 \leq r_2$. The range R is the set of all bit strings x of W bits such that $x \in [r_1, r_2]$. A bit string x of W bits is said to *match* the range (or be in the range) R if $x \in [r_1, r_2]$.

In particular, a range R is a *prefix range*, with a *prefix* $r' \in \{0,1\}^k$ of length $k \in [0, W]$ if $r_1 = r'(0)^{W-k}$, and $r_2 = r'(1)^{W-k}$. It is a single point or an *exact match* if $r_1 = r_2$.

We define the sets of extremal ranges and generalized extremal ranges, which are subsets of all ranges.

Definition 2 (Extremal Range, Generalized Extremal Range): A range $R = [r_1, r_2] \subseteq [0, 2^W - 1]$ of W-bit binary strings is called an *extremal range* when $r_1 = 0$ or $r_2 = 2^W - 1$.

In addition, let $R' = [r'(0)^{W-k}, r'(1)^{W-k}]$ be the minimal-size prefix range that contains R, i.e., satisfies $R \subseteq R'$. We say that the range R is a generalized extremal range if $r_1 = r'(0)^{W-k}$ or $r_2 = r'(1)^{W-k}$.

Example 1: As mentioned above, let $W = 6$ and consider the range $R = [r_1, r_2] = [5, 7] = \{000101, \ldots, 000111\}$. The range R is contained in the prefix range $R' = [4, 7] = [r'(0)^{W-2}, r'(1)^{W-2}]$ for $r' = 0001$. Since $r_2 = 7 = r'(1)^{W-2}$, we say that the range $R = [r_1, r_2]$ is a generalized extremal range. Since $r_1 \neq 0$ and $r_2 \neq 2^W - 1 = 63$, the range R is not an extremal range.

Definition 3 (TCAM Entry, Prefix TCAM Entry): A *TCAM entry* S of width W is a ternary string $S = s_1 \ldots s_W \in \{0, 1, *\}^W$, where $\{0, 1\}$ are bit values and $*$ stands for *don't-care*. A W-bit string $b = b_1 \ldots b_W$ matches S, denoted as $b \in S$, if and only if for all $i \in [1, W]$, $s_i \in \{b_i, *\}$. We use S to denote also the set of strings that it matches, when no confusion will arise.

A TCAM entry $S = s_1 \ldots s_W \in \{0, 1, *\}^W$ is a prefix TCAM entry if $s_j = *$ for some $j \in [1, W]$ implies that $s_{j'} = *$ for any $j' \in [j, W]$.

Note that prefix TCAM entries of width W are in one-to-one correspondence with prefix ranges of width W. A range with a prefix r corresponds to the prefix TCAM entry $r(*)^{W-k}$.

We assume that each TCAM entry S is associated with an indication a that is either *in* or *out*. We denote a pair consisting of an entry S and an indication a by $S \rightarrow a$. Depending on the context, we shall refer by a *TCAM entry* either to S or to the pair $S \rightarrow a$.

To simplify our presentation, we assume at first that the packet header consists of a single field of width W. We focus on a single classification rule defined by a general range over this field and a corresponding action that should be applied on bit strings in the range. We call such a rule a *range rule*. Later, we also discuss headers with two fields of width W each, in which case the width of the header and of the TCAM entries would be $2W$.

Definition 4 (TCAM Encoding of a Range): A *TCAM encoding* ϕ of a range R of width W is a sequence of TCAM entries $(S_1 \rightarrow a_1, \ldots, S_n \rightarrow a_n)$ where each a_i is either *in* or *out*. This sequence satisfies that for each header $x \in \{0,1\}^W$ such that $x \in R$, the first TCAM entry S_j matching x is associated with $a_j = in$; for each $x \notin R$, either the first TCAM entry S_j matching x is associated with $a_j = out$, or no TCAM entry matches x. (we assume a default indication of *out*). The number of rules, n, is called *the size of* ϕ and denoted by $|\phi|$.

A *prefix TCAM encoding* ϕ of a range R is a TCAM encoding of R in which all entries are prefix TCAM entries.

B. Optimal Range Encoding Schemes

For each range R, we denote by $OPT(R)$ a smallest TCAM encoding of R, and by $OPT_p(R)$ a smallest prefix TCAM encoding of R. We also denote $opt(R) = |OPT(R)|$ and $opt_p(R) = |OPT_p(R)|$. We refer to $opt(R)$ as the *range expansion* of R, or just the *expansion* of R for short. Likewise, we refer to $opt_p(R)$ as the *prefix range expansion* of R, or just the *prefix expansion* of R for short.

We define $r(W)$ to be the maximum expansion of a range in $\{0,1\}^W$, that is $r(W) = \max_R opt(R)$. Similarly, we define $r^e(W)$ to be the maximum expansion of an extremal range, that is $r^e(W) = \max\{opt(R) | R = [0,y] \vee R = [y, 2^W - 1]\}$. Analogously, we define the maximum expansion with prefix TCAM entries to be $r_p(W) = \max_R opt_p(R)$, and for extremal ranges $r_p^e(W) = \max\{opt_p(R) | R = [0,y] \vee R = [y, 2^W - 1]\}$.

Our main goal is to find an algorithm that encodes a range R with $opt(R)$ rules and to understand the expected value of $opt(R)$ over all ranges. Another goal is to find $r(W)$, $r^e(W)$, $r_p(W)$, and $r_p^e(W)$.

III. In/Out TCAM Architecture

In this section, we describe the new *In/Out TCAM architecture*. The architecture enables independent encoding of each rule in a classifier without considering the possible dependencies between the rules. It also reduces the expansion of hard-to-encode rules by encoding them with a modified TCAM, using both *in* and *out* entries.

As Fig. 1 illustrates, the architecture combines a regular TCAM and a modified TCAM. The regular TCAM encodes the simple rules. The modified TCAM encodes the hard-to-encode rules (e.g., the two-dimensional or the nonextremal one-dimensional rules) with a *guaranteed improved expansion*.

Each incoming header is sent to both TCAMs. Each TCAM outputs the index of the first rule (among those that it encodes) that the header matches. The outputs of the two TCAMs get into a *conversion module* that spits out the corresponding action. This conversion module has to choose the rule to apply from its two inputs and convert it into an action (as an ordinary conversion module does). It can choose the applicable rule among its inputs by picking the one with the smaller index assuming these indices had been converted to global indices in some list containing all rules.

In order to output the first rule that the header matches, the modified TCAM uses a two-level logic. The first level gets as input the match/unmatch bit of each TCAM entry, and it outputs a match/unmatch bit for each rule encoded by the modified TCAM. The second level is just a regular priority encoder that outputs the first among the rules (encoded by the modified TCAM) that the header matches.

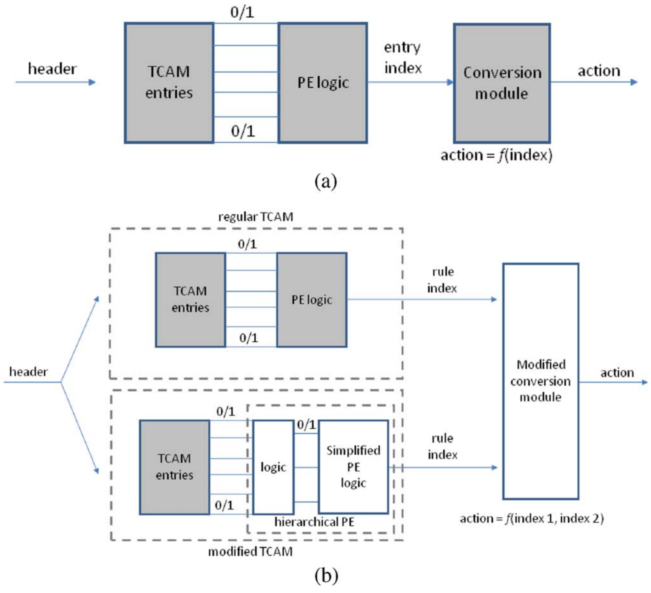


Fig. 1. Comparison of a regular TCAM architecture with the suggested TCAM architecture. Components that also appear in the regular TCAM are presented in gray. (a) Regular TCAM architecture. A priority encoder (PE) is used to select the first matching entry. Then, an action is selected based on the entry index. (b) Suggested In/Out TCAM architecture. It includes a regular TCAM and a modified TCAM. In the modified TCAM, each range is encoded separately, and the regular PE is replaced by a hierarchical PE that is used to select the first matching range. Finally, the action is selected based on the indices sent by the two TCAMs.

The logic of the first level can be implemented as follows. The entries of the modified TCAM are partitioned into groups $G_1, \ldots, G_m$, where group G_i includes all the entries encoding rule i, $i = 1, \ldots, m$, and m is the number of rules encoded by the modified TCAM. For each group G_i, we have a priority encoder PE_i that gets the match/unmatch bits of the entries of G_i and outputs the index g_i of the first matching entry among G_i. We then use this index to access a table S_i. This table stores a 0 at a position of an *out* entry and a 1 at a position of an *in* entry. The output of the first-level logic for rule i is the bit $S_i[g_i]$. It follows that the header matches rule i if and only if its first matching entry in G_i is an *in* entry.

The size of the additional logic is small—it uses a small constant number of gates for each entry in the modified TCAM—since the size of PE_i is linear in the size of the group G_i that it encodes. This adds a small constant factor to the size and the delay of the priority encoder attached to a regular TCAM.

The additional logic used by the modified TCAM is not an off-the-shelf component. Furthermore, the structure of this additional logic depends on the number and the expansions of the specific rules that are encoded by the modified TCAM. The additional logic is simple and always consists of a group of priority encoders PE_i and associated tables S_i, but the number of inputs of each PE_i and the size of S_i depends on the expansion of rule i. Furthermore, the contents of S_i depend on the classification of the entries of G_i as *in* or *out*. We can implement the additional logic using some programmable device (such devices are common and cheap today) that we reprogram each time we change the rules that our modified TCAM encodes.

We can simplify the structure of the additional logic at the expense of wasting some space of the modified TCAM, if we allocate the same fixed number of entries, say L to each rule. That is, we make $|G_i| = L$ for all i. This allows us to use the same implementation of the additional logic as long as we do not change L (even if we do change the rules themselves). This implementation should be designed such that it can get the contents of the tables S_i (which are now of size L each) as input.

This new architecture is especially interesting because the fraction of complex classification rules is increasing and is expected to increase dramatically with the introduction of virtualization and the related flexible flow matching in software-defined networking (SDN) [10], [33]. Therefore, this new architecture may help solve future scaling bottlenecks since it provides tight guarantees on the worst-case number of entries needed for each rule.

The suggested architecture can also improve the worst-case *power consumption* of the conventional TCAM, known to be roughly proportional to the number of TCAM entries [17]. By encoding the two-dimensional rules in the modified TCAM, we can significantly reduce the number of entries (e.g., by improving for $W = 16$ the maximum expansion from $(2W-2)^2 = 900$ to $2W = 32$). Although the In/Out TCAM architecture includes two TCAMs, a regular one and a modified one with a more complicated logic, we believe that the gain from reducing the number of entries is more significant and the new architecture will require less power.

IV. Extremal 1-D Ranges

In this section, we consider the expansion of one-dimensional extremal ranges over the set of prefix encoding schemes and over the set of all encoding schemes.

For $y \in [0, 2^W - 1]$, an extremal range may be a left-extremal range of the form $R^{LE} = [0, y]$ or a right-extremal range of the form $R^{RE} = [y, 2^W - 1]$.

Given a TCAM encoding scheme ϕ that encodes a left-extremal range $R = [0, y]$ with $|\phi|$ TCAM entries, we can obtain a TCAM encoding scheme ϕ' that encodes the right-extremal range $R' = [2^W - 1 - y, 2^W - 1]$ in exactly $|\phi|$ TCAM entries. To do so, invert each of the bit values 0 and 1 (and ignore the don't-cares) in all the $|\phi|$ entries. Hence, the range expansion of a right-extremal range is the same as the range expansion of a corresponding left-extremal range, and it suffices to consider only left-extremal ranges.

Note that while we deal with *extremal* ranges, the results below also apply to *generalized extremal* ranges. This is because each generalized extremal range is simply an extremal range in a smaller range (smaller W) defined by its subtree. (we simply ignore the fixed sequence of most significant bits in the binary representations of all the values in the range.) Therefore, for simplicity, we consider extremal ranges.

A. Prefix Encoding Versus General Encoding of Extremal Ranges

The next theorem compares, for *any* extremal range R, the size of the smallest TCAM encoding of R and the size of the smallest prefix TCAM encoding of R. It shows that they are actually identical.

Theorem 1: For any extremal range $R = [0, y]$ (where $y \in [0, 2^W - 1]$), the prefix range expansion of R is exactly the range expansion of R, i.e.,

$$opt_p(R) = opt(R). \tag{1}$$

Proof: We consider an arbitrary extremal range $R = [0, y] = \{(0)^W, \ldots, y_1 \ldots y_W\}$ and want to show that $opt_p(R) = opt(R)$.

We trivially have that $opt_p(R) \geq opt(R)$, and we only have to prove that $opt_p(R) \leq opt(R)$. Consider all minimal encoding schemes of R. Among them, consider the schemes with the smallest number of non-prefix entries and, in this subset, the schemes with the smallest number of $*$'s in their non-prefix entries. Let $\phi = (S_1 \to a_1, \ldots, S_n \to a_n)$ be such a minimal encoding scheme. We show that we can encode R in a prefix encoding scheme with at most $|\phi|$ entries.

If all the TCAM entries of ϕ are prefix TCAM entries, we are done. Thus, we assume that ϕ has at least one non-prefix TCAM entries.

Among the non-prefix TCAM entries of ϕ, we look at the index of the leftmost $*$ in each entry. We then consider the entry with the smallest index of its leftmost $*$. If there are several non-prefix entries with the same index of their lef-most $*$, we consider the last one. We denote this entry by $S \to a$ such that $S = (s_1, \ldots, s_W) \in \{0, 1, *\}^W$ and distinguish two different cases depending on whether the indication a is *in* or *out*. Let k be the index of this TCAM entry that is $(S \to a) = (S_k \to a_k)$, and let $j \in [1, W]$ be the minimal index such that $s_j = *$.

We first consider the case where $a =$ in. The case $a =$ out is similar, and we discuss it shortly at the end of the proof. We compare the first (leftmost) $j-1$ symbols of y, the right endpoint of our range $R = [0, y]$, and S. By the definition of j, we have that $\forall i \in [1, (j-1)]$, $s_i \in \{0, 1\}$, and therefore $y_1 \ldots y_{j-1}$ and $s_1 \ldots s_{j-1}$ are both binary strings. The proof now splits into several cases.

1) We have $s_1 \ldots s_{j-1} > y_1 \ldots y_{j-1}$. In this case, the entry $S_k \to$ in positively matches strings that are not in the range, and therefore these strings must match preceding *out* entries. It follows that we can remove the entry $S_k \to$ in and get a smaller encoding of R. This is a contradiction to the minimality of ϕ.
2) We have $s_1 \ldots s_{j-1} < y_1 \ldots y_{j-1}$. In this case, one can replace $S_k \to$ in with $s_1 \ldots s_{j-1}(*)^{W-j+1} \to$ in to get an encoding of R with a smaller number of non-prefix entries, which contradicts the definition of ϕ.
3) We have $s_1 \ldots s_{j-1} = y_1 \ldots y_{j-1}$ and $y_j = 0$. In this case, one can replace $S_k \to$ in with $s_1 \ldots s_{j-1} 0 s_{j+1} \ldots s_W \to$ in to get an encoding of R in which we have one less $*$ in non-prefix entries in contradiction to the definition of ϕ.
4) We have $s_1 \ldots s_{j-1} = y_1 \ldots y_{j-1}$, $y_j = 1$, and there exists an entry $S_\ell \to a_\ell$ that begins with $s_1 \ldots s_{j-1} 0$. If $a_\ell = out$, then by deleting the entry $S_\ell \to a_\ell$, we get a smaller encoding of R. If a_ℓ is *in*, change the encoding as follows: Remove the entry $S_\ell \to a_\ell$, change $S_k \to a_k$ to $s_1 \ldots s_{j-1} 1 s_{j+1} \ldots s_W \to a_k$, and add the entry $s_1 \ldots s_{j-1} 0 (*)^{W-j} \to$ in as a first entry. This either gives an encoding of R with fewer non-prefix entries or an encoding of R with fewer $*$'s in non-prefix entries, which contradicts the definition of ϕ.
5) Finally, we have $s_1 \ldots s_{j-1} = y_1 \ldots y_{j-1}$, $y_j = 1$, and there is no entry in the encoding that begins with

$s_1 \dots s_{j-1}0$. Let B denote the set of 2^{W-j} strings that begin with $s_1 \dots s_{j-1}0$. Let us assume first that there is an entry in $S_{k+1} \to a_{k+1}, \dots, S_n \to a_n$ that is matched by at least one of the strings in B. Let $S_\ell \to a$ be the first such entry. Since there is no entry that begins with $s_1 \dots s_{j-1}0$, it must be that the index of the leftmost $*$ in S_ℓ is at most j. Since the entry $S_k \to a_k$ is the non-prefix entry with the leftmost $*$, and the last non-prefix entry among all the non-prefix entries with $*$ in position j, it follows that S_ℓ is a prefix entry, of the form $s_1 \dots s_r(*)^{W-r}$ with $r < j$. This means that for every string in B, and in particular for the strings in B that are first matched by S_k, the first matching entry in $S_{k+1} \to a_{k+1}, \dots, S_n \to a_n$ is $S_\ell \to a_\ell$.

If $a_\ell = in$, then we can change $S_k \to in$ to be $s_1 \dots s_{j-1}1s_{j+1} \dots s_W \to in$ and get an encoding of R with one less $*$ in non-prefix entries—all the strings in B that were positively matched by $S_k \to in$ are positively matched by $S_\ell \to in$ instead.

If $a_\ell = out$ or if there is no entry in $S_{k+1} \to a_{k+1}, \dots, S_n \to a_n$ that is matched by a string in B, then $S_1 \to a_1, \dots, S_k \to a_k$ positively match all the strings in B. Since there is no entry (anywhere) that begins with $s_1 \dots s_{j-1}0$, it must be the case that every string x that begins with $s_1 \dots s_{j-1}1$ is also matched by one of the first k entries. Therefore, if $x \notin R$, then when reaching the kth entry, x is already negatively encoded. This means that we can change S_k to be $s_1 \dots s_{j-1}(*)^{W-j+1} \to \text{in}$ while still encoding R. This decreases the number of non-prefix entries in the encoding and contradicts the definition of ϕ.

The case $a = a_k = out$ is similar: If we replace the indications *in* and *out* and change the default indication (for strings that are not matched by any rule) from *out* to *in*, then we get a minimal encoding of the complement of R with the smallest number of non-prefix rules and the smallest number of $*$'s in these rules. The kth rule, which is the non-prefix rule with the leftmost $*$ (and the last among those if there is more than one), is now an *in* rule. We then apply an argument analogous to the above and get a contradiction. Note that we can slightly modify case 5) so that it still applies despite the fact that we change the default indication from *out* to *in*. □

B. Optimal Encoding Scheme for Any Given Extremal Range

In this section, we present an algorithm that computes, for any given extremal range R, *an optimal encoding of* R. By Theorem 1, it is sufficient to find the optimal encoding with prefix TCAM entries.

Let T be a subtree of the binary tree (with 2^W leaves) describing the entire space $[0, 2^W - 1]$. The subtree T corresponds to all binary strings starting with a particular prefix $x(T)$. That is, the subtree T consists of all the strings matching the TCAM entry $c(T) = x(T)(*)^{W-|x(T)|}$. Given a range $R \subseteq [0, 2^W - 1]$, and a subtree T, we call a prefix TCAM encoding of $R \cap T$, such that all of its entries start with $x(T)$, *a prefix TCAM encoding of* $R \cap T$ *within* T.

For a subtree T, we define $IN^T(R \cap T)$ to be a shortest prefix TCAM encoding of $R \cap T$ within T in which the last entry is of the form $c(T) \to \text{in}$, and let $n_{IN}^T(R \cap T)$ be the number of entries in $IN^T(R \cap T)$. (If the shortest such encoding is not unique, then $IN^T(R \cap T)$ is an arbitrary one of them.)

Similarly, let $OUT^T(R \cap T)$ be a shortest prefix TCAM encoding of $R \cap T$ within T in which the last entry is $c(T) \to \text{out}$, and let $n_{OUT}^T(R \cap T)$ be the number of entries in $OUT^T(R \cap T)$. In the following, we typically omit the superscript T, which will be clear from the context.

Example 2: If a subtree T satisfies $T \subseteq R$, then $R \cap T = T$ can be encoded by $IN(R \cap T) = (c(T) \to \text{in})$ in $n_{IN}(R \cap T) = 1$ entries or by $OUT(R \cap T) = (c(T) \to \text{in}, c(T) \to \text{out})$ in $n_{OUT}(R \cap T) = 2$ entries. If $T \subseteq R^c$, then $R \cap T$ can be encoded by $IN(R \cap T) = (c(T) \to \text{out}, c(T) \to \text{in})$ in $n_{IN}(R \cap T) = 2$ entries or by $OUT(R \cap T) = (c(T) \to \text{out})$ in $n_{OUT}(R \cap T) = 1$ entries.

If the subtree T is a leaf and thereby contains a single string, then either $T \subseteq R$ or $T \subseteq R^c$. Thus, $IN(R \cap T)$, $n_{IN}(R \cap T)$, $OUT(R \cap T)$, and $n_{OUT}(R \cap T)$ can be computed as in Example 2. In preparation for our dynamic programming algorithm, we state the following propositions whose straightforward proofs can be found in [34]. The first proposition shows how we can compute $IN(R \cap T)$, $n_{IN}(R \cap T)$, $OUT(R \cap T)$, and $n_{OUT}(R \cap T)$ for $|T| \geq 2$ based on the corresponding value for the left and the right subtrees of T denoted by $\ell(T)$ and $r(T)$, respectively.

Proposition 1: Let T be a subtree such that $|T| \geq 2$. Let $\ell(T)$ and $r(T)$ be the left and the right subtrees of T, respectively. Then

$$
\begin{aligned}
&n_{IN}^T(R \cap T) \\
&\quad = \min\Big\{ n_{IN}^{\ell(T)}(R \cap \ell(T)) + n_{IN}^{r(T)}(R \cap r(T)) - 1, \\
&\qquad\qquad n_{OUT}^{\ell(T)}(R \cap \ell(T)) + n_{OUT}^{r(T)}(R \cap r(T)) \Big\} \\
&n_{OUT}^T(R \cap T) \\
&\quad = \min\Big\{ n_{IN}^{\ell(T)}(R \cap \ell(T)) + n_{IN}^{r(T)}(R \cap r(T)), \\
&\qquad\qquad n_{OUT}^{\ell(T)}(R \cap \ell(T)) + n_{OUT}^{r(T)}(R \cap r(T)) - 1 \Big\}.
\end{aligned}
$$

The following proposition relates the values of $opt_p(R)$ and $n_{OUT}(R \cap T)$ for the complete tree T describing the entire space $[0, 2^W - 1]$.

Proposition 2: Let T be the complete binary tree of the range $[0, 2^W - 1]$ (i.e., $c(T) = (*)^W$). The prefix range expansion of a range R is $n_{OUT}(R \cap T) - 1$, i.e., $opt_p(R) = n_{OUT}(R \cap T) - 1$.

Our third proposition is the following.

Proposition 3: For any subtree T, $n_{OUT}(R \cap T) \leq n_{IN}(R \cap T) + 1$ and $n_{IN}(R \cap T) \leq n_{OUT}(R \cap T) + 1$. That is, we have $|n_{IN}(R \cap T) - n_{OUT}(R \cap T)| \leq 1$.

Based on Proposition 1, we describe a simplified version of a dynamic-programming algorithm presented in [13] to compute an optimal encoding of any extremal range. Our algorithm is faster and simpler since we only consider the $W + 1$ subtrees $T_i = y_1 \dots y_{W-i}(*)^i$ (for $i \in [0, W]$). (We note that the algorithm in [13] is more general and can deal with several ranges that are not necessarily extremal and defined on one or two dimensions.) In each step of the algorithm, we compute $n_{IN}(R \cap T)$ and $n_{OUT}(R \cap T)$ from $n_{IN}(R \cap \ell(T))$, $n_{IN}(R \cap r(T))$, $n_{OUT}(R \cap \ell(T))$, and $n_{OUT}(R \cap r(T))$, where either $n_{IN}(R \cap \ell(T))$ and $n_{OUT}(R \cap \ell(T))$ or $n_{IN}(R \cap r(T))$ and $n_{OUT}(R \cap r(T))$ are obtained immediately as in Example 2.

We recall that, by Theorem 1, the encoding that we compute is optimal among all encoding schemes rather than just among prefix schemes.

Algorithm 1: Consider an arbitrary extremal range $R = [0, y] = \{(0)^W, \ldots, y_1 \ldots y_W\}$. To optimally encode it, we first compute $IN(R \cap T), OUT(R \cap T)$, $n_{IN}(R \cap T), n_{OUT}(R \cap T)$ for the $W+1$ different subtrees $T_0, T_1, \ldots, T_W$ where $c(T_j) = y_1 \ldots y_{W-j}(*)^j$. Each subtree is rooted at a different level of the complete binary tree of the range $[0, 2^W - 1]$, T_0 is a single leaf, and T_W is the entire complete binary tree. By Proposition 2, an optimal encoding of R is given by the $n_{OUT}(R \cap T_W) - 1$ first entries of $OUT(R \cap T_W)$.

Since $T_0 \subseteq R = [0, y] = \{(0)^W, \ldots, y_1 \ldots y_W\}$, we have that $IN(R \cap T_0) = (c(T_0) \to \text{in})$ and $n_{IN}(R \cap T_0) = 1$. Similarly, $OUT(R \cap T_0) = (c(T_0) \to \text{in}, c(T_0) \to \text{out})$ and $n_{OUT}(R \cap T_0) = 2$, as described in Example 2.

Now we assume that we have already computed $IN(R \cap T_{i-1})$, $n_{IN}(R \cap T_{i-1})$, $OUT(R \cap T_{i-1})$, and $n_{OUT}(R \cap T_{i-1})$ and show how to compute $IN(R \cap T_i)$, $n_{IN}(R \cap T_i)$, $OUT(R \cap T_i)$, and $n_{OUT}(R \cap T_i)$.

If $y_{W-i+1} = 0$, then $\ell(T_i) = T_{i-1}$ and $r(T_i) \subseteq R^c$.

We can obtain $OUT(R \cap T_i)$ from $OUT(R \cap T_{i-1})$ by replacing its last entry $c(T_{i-1}) \to$ out with $c(T_i) \to$ out so $n_{OUT}(T_i) = n_{OUT}(T_{i-1})$.[1]

To compute $IN(R \cap T_i)$ and $n_{IN}(R \cap T_i)$, we first note that since $r(T_i) \subseteq R^c$ (as in Example 2), we have that $IN(R \cap r(T_i)) = (c(r(T_i)) \to \text{out}, c(r(T_i)) \to \text{in})$, $n_{IN}(R \cap r(T_i)) = 2$, and $OUT(R \cap r(T_i)) = (c(R_{T_i}) \to \text{out})$, $n_{OUT}(R \cap r(T_i)) = 1$. Thus, from Proposition 1, it follows that

$$\begin{aligned} &n_{IN}(R \cap T_i) \\ &\quad = \min\{n_{IN}(R \cap \ell(T_i)) + n_{IN}(R \cap r(T_i)) - 1, \\ &\qquad\qquad n_{OUT}(R \cap \ell(T_i)) + n_{OUT}(R \cap r(T_i))\} \\ &\quad = \min\{n_{IN}(R \cap \ell(T_i)) + 1, n_{OUT}(R \cap \ell(T_i)) + 1\}. \quad (2) \end{aligned}$$

We now split into two subcases according to the values of $n_{IN}(R \cap \ell(T_i))$ and $n_{OUT}(R \cap \ell(T_i))$. By Proposition 3, these are the only subcases possible.

*Subcase 1:*If $n_{IN}(R \cap \ell(T_i)) + 1 = n_{OUT}(R \cap \ell(T_i))$ or $n_{IN}(R \cap \ell(T_i)) = n_{OUT}(R \cap \ell(T_i))$, then by (2), we get that $n_{IN}(R \cap T_i) = \min\{n_{IN}(R \cap \ell(T_i)) + 1, n_{OUT}(R \cap \ell(T_i)) + 1\} = n_{IN}(R \cap \ell(T_i)) + 1$.

We can get $IN(R \cap T_i)$ by replacing the last entry $c(\ell(T_i)) \to$ in of $IN(R \cap \ell(T_i))$ by the two entries $(c(r(T_i)) \to \text{out}, c(T_i) \to \text{in})$.

*Subcase 2:*If $n_{IN}(R \cap \ell(T_i)) = n_{OUT}(R \cap \ell(T_i)) + 1$, then by (2), we get that $n_{IN}(R \cap T_i) = \min\{n_{IN}(R \cap \ell(T_i)) + 1, n_{OUT}(R \cap \ell(T_i)) + 1\} = n_{OUT}(R \cap \ell(T_i)) + 1 = n_{IN}(R \cap \ell(T_i))$. To get $IN(R \cap T_i)$, we replace the last entry $c(\ell(T_i)) \to$ out of $OUT(R \cap \ell(T_i))$ by the two entries $(c(T_i) \to \text{out}, c(T_i) \to \text{in})$.

If $y_{W-i+1} = 1$, then $\ell(T_i) \subseteq R$ and $r(T_i) = T_{i-1}$. The analysis is symmetric to the case $y_{W-i+1} = 0$ and goes as follows.

We can obtain $IN(R \cap T_i)$ from $IN(R \cap T_{i-1})$ by replacing its last entry $c(T_{i-1}) \to$ in by $c(T_i) \to$ in so $n_{IN}(R \cap T_i) = n_{IN}(R \cap T_{i-1})$.

As in Example 2, $IN(R \cap \ell(T_i)) = (c(\ell(T_i)) \to \text{in})$, $n_{IN}(R \cap \ell(T_i)) = 1$, $OUT(R \cap \ell(T_i)) = (c(\ell(T_i)) \to \text{in}, c(\ell(T_i)) \to \text{out})$, and $n_{OUT}(R \cap \ell(T_i)) = 2$. Thus, from Proposition 1, it follows that

$$\begin{aligned} &n_{OUT}(R \cap T_i) \\ &\quad = \min\{n_{IN}(R \cap \ell(T_i)) + n_{IN}(R \cap r(T_i)), \\ &\qquad\qquad n_{OUT}(R \cap \ell(T_i)) + n_{OUT}(R \cap r(T_i)) - 1\} \\ &\quad = \min\{n_{IN}(R \cap r(T_i)) + 1, n_{OUT}(R \cap r(T_i)) + 1\}. \quad (3) \end{aligned}$$

We compute $OUT(T_i)$ and $n_{OUT}(T_i)$ based on the values $n_{IN}(R_{T_i})$ and $n_{OUT}(R_{T_i})$ by the appropriate of the following subcases.

Subcase 1: If $n_{IN}(R \cap r(T_i)) = n_{OUT}(R \cap r(T_i)) + 1$ or $n_{IN}(R \cap r(T_i)) = n_{OUT}(R \cap r(T_i))$, then by (3), $n_{OUT}(R \cap T_i) = \min\{n_{IN}(R \cap r(T_i)) + 1, n_{OUT}(R \cap r(T_i)) + 1\} = n_{OUT}(R \cap r(T_i)) + 1$. We can get $OUT(R \cap T_i)$, and we replace the last entry $c(r(T_i)) \to$ out of $OUT(R \cap r(T_i))$ by the two entries $(c(\ell(T_i)) \to \text{in}, c(T_i) \to \text{out})$.

Subcase 2: If $n_{IN}(R \cap r(T_i)) + 1 = n_{OUT}(R \cap r(T_i))$, then by (3), $n_{OUT}(R \cap T_i) = \min\{n_{IN}(R \cap r(T_i)) + 1, n_{OUT}(R \cap r(T_i)) + 1\} = n_{IN}(R \cap r(T_i)) + 1 = n_{OUT}(R \cap r(T_i))$. To get $OUT(R \cap T_i)$, we replace the last entry $c(r(T_i)) \to$ in of $IN(R \cap r(T_i))$ by the two entries $(c(T_i) \to \text{in}, c(T_i) \to \text{out})$.

Example 3: Fig. 2(a) illustrates the results of the algorithm for the range $R = [0, 22] = \{(0)^W, \ldots, y_1 \ldots y_W\}$ for $W = 5$ and $y_1 \ldots y_W = 10110$. First, for $T_0 = \{y_1 \ldots y_W\}$, we clearly have $n_{IN}(R \cap T_0) = 1$ and $n_{OUT}(R \cap T_0) = 2$. Similarly, for $i \in [1, W]$, the values $n_{IN}(R \cap T_i)$ and $n_{OUT}(R \cap T_i)$ of the subtree T_i where $c(T_i) = y_1 \ldots y_{W-i}(*)^i$ are also presented. By Proposition 2, $opt(R) = opt_p(R) = n_{OUT}(R \cap T_W) - 1 = 4 - 1 = 3$ and R can be encoded as $(10111 \to \text{out}, 11*** \to \text{out}, ***** \to \text{in})$.

C. Range Expansion of a Given Extremal Range

We derive from our algorithm a simple *deterministic finite automata* (DFA) that computes the optimal range expansion of a given extremal range $R = [0, y] = \{(0)^W, \ldots, y_1 \ldots y_W\}$. This automata will be useful for analyzing the expected range expansion over all extremal ranges.

The DFA, shown in Fig. 2(b), consists of three states $Q = \{A, B, C\}$. These three states represent the three possible values of $n_{IN}(R \cap T) - n_{OUT}(R \cap T) \in \{-1, 0, 1\}$ for a subtree T, in a way that we make precise in Proposition 4. The state $A = (a, a+1)$ represents a subtree T with $n_{IN}(R \cap T) + 1 = n_{OUT}(R \cap T)$, the state $B = (b, b)$ represents a subtree T with $n_{IN}(R \cap T) = n_{OUT}(R \cap T)$, and the state $C = (c+1, c)$ represents a subtree T with $n_{IN}(R \cap T) = n_{OUT}(R \cap T) + 1$.

The input to the DFA is the binary string $y_1 \ldots y_W$ *in a right-to-left order*. The starting state is A, and the transition function $\delta : Q \times \Sigma \to Q$ is defined such that $\delta(A, 0) = B$, $\delta(A, 1) = A$, $\delta(B, 0) = C$, $\delta(B, 1) = A$, $\delta(C, 0) = C$, and $\delta(C, 1) = B$. (Since we are not interested in the language this DFA accepts, we do not define accepting states.)

We want to show how to derive the expansion of $R = \{(0)^W, \ldots, y_1 \ldots y_W\}$ from the computation of this DFA on $y_W \ldots y_1$. To do so, we define the state $q_i \in Q$, for $i \in [0, W]$, to be the state of the DFA after reading the first i input bits $y_W, \ldots, y_{W-i+1}$ and use the following proposition.

Proposition 4: Let T_i be the subtree corresponding to the set $y_1 \ldots y_{W-i}(*)^i$. The state q_i corresponds to the values of

[1]It is easy to see that $OUT(R \cap T_i)$ is not shorter than $OUT(R \cap T_{i-1})$.

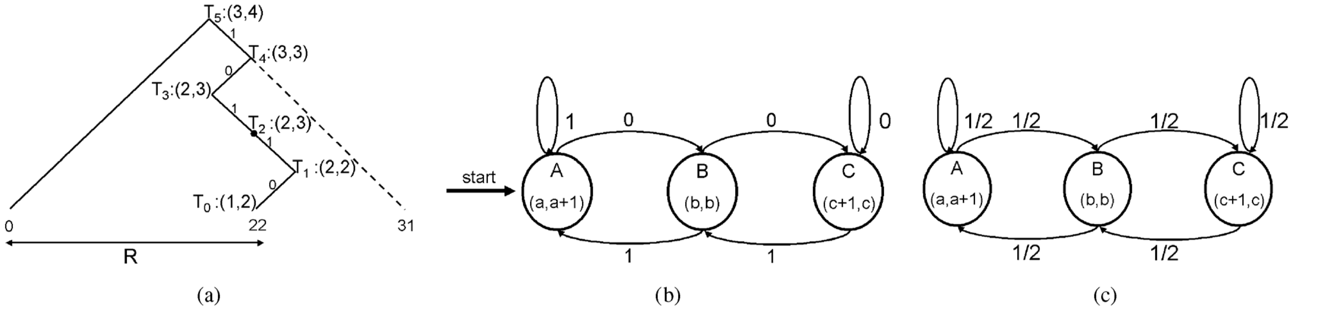


Fig. 2. (a) Illustration of the algorithm results for the extremal range $R = [0, 22]$ from Example 3. The values $(n_{IN}(R \cap T_i), n_{OUT}(R \cap T_i))$ of each tree $T_i \in \{T_0, \ldots, T_W\}$ are illustrated. The parameter $n_{IN}(R \cap T_i)$ is the number of entries in the smallest encoding of $R \cap T_i$ with entries that start with $x(T)$ and with a last entry of the form $c(T_i) \to$ in. Likewise, $n_{OUT}(R \cap T_i)$ is the size of the smallest encoding with entries that start with $x(T)$ and with a last entry of the form $c(T_i) \to$ out. The smallest encoding of R has $opt(R) = n_{OUT}(R \cap T_5) - 1 = 4 - 1 = 3$ entries. (b) DFA, as discussed in Section IV-C. It has three states representing the three possible values of $(n_{IN}(R \cap T) - n_{OUT}(R \cap T)) \in \{-1, 0, 1\}$ in a subtree T. (c) Corresponding Markov Chain of the DFA with the same 3 states.

$n_{IN}(R \cap T_i)$ and $n_{OUT}(R \cap T_i)$ as follows. If $q_i = A = (a, a+1)$, then $n_{IN}(R \cap T_i) + 1 = n_{OUT}(R \cap T_i)$. If $q_i = B = (b, b)$, then $n_{IN}(R \cap T_i) = n_{OUT}(R \cap T_i)$, and if $q_i = C = (c+1, c)$, then $n_{IN}(R \cap T_i) = n_{OUT}(R \cap T_i) + 1$.

Proof: The proof is by induction on i. For $i = 0$, $q_0 = A = (a, a+1)$ and indeed $(n_{IN}(T_0), n_{OUT}(T_0)) = (1, 2)$ as explained in Example 2.

The induction step follows from the previous description of the recursive formulas for $n_{IN}(T_i)$, and $n_{OUT}(T_i)$. For example, assume that $q_i = A = (a, a+1)$; that is, by induction, we have that $n_{IN}(T_i) + 1 = n_{OUT}(T_i)$. If the $(i+1)$th symbol, y_{W-i}, processed by the DFA, is 0, then $n_{IN}(T_{i+1}) = n_{IN}(T_i) + 1 = n_{OUT}(T_i) = n_{OUT}(T_{i+1})$, and indeed we have $\delta(A, 0) = B = (b, b)$, so $q_{i+1} = B$ as required. If $y_{W-i} = 1$, then $n_{IN}(T_{i+1}) + 1 = n_{IN}(T_i) + 1 = n_{OUT}(T_i) = n_{OUT}(T_{i+1})$, and since $\delta(A, 1) = A$, we have that $q_{i+1} = A = (a, a+1)$ as required. Similarly, we can show the correctness of the induction step for the four other transitions. $\square$

The next theorem explains how we can obtain the expansion of the range $R = [0, y] = \{(0)^W, \ldots, y_1 \ldots y_W\}$ from the transitions of the DFA while processing $y_W, \ldots, y_1$.

Theorem 2: Let n_y be the number of transitions of the form $\delta(B, 1) = A$ or $\delta(C, 1) = B$ that the DFA makes while processing $y_W, \ldots, y_1$. Then, the range expansion of the extremal range $R = [0, y] = \{(0)^W, \ldots, y_1 \ldots y_W\}$ satisfies $opt(R) = n_y + 1$.

Proof: For $i \in [0, W]$, let T_i be the subtree corresponding to $y_1 \ldots y_{W-i}(*)^i$ as before. Furthermore, let n_i be the number of transitions of the form $\delta(B, 1) = A$ or $\delta(C, 1) = B$ that the DFA makes while processing $y_W \ldots y_{W-i+1}$. We show below by induction on i that $n_{OUT}(T_i) = n_i + 2$, for $i \in [0, W]$. Then, from Proposition 2, we get that $opt_p(R) = n_{OUT}(T_W) - 1 = n_W + 2 - 1 = n_W + 1 = n_y + 1$. Finally, by Theorem 1, $opt(R) = opt_p(R) = n_y + 1$, and the theorem follows.

Now the induction showing that $n_{OUT}(T_i) = n_i + 2$ goes as follows. First, $n_{OUT}(T_0) = 2$ as discussed before, and $n_0 = 0$ since the DFA has not yet processed any symbol. For the induction step, we observe that by the definition of n_i, $n_{i+1} = n_i + 1$ if the $(i+1)$th transition is of the form $\delta(B, 1) = A$ or $\delta(C, 1) = B$, and $n_{i+1} = n_i$ otherwise. By the proof of Proposition 4, $n_{OUT}(T_{i+1}) = n_{OUT}(T_i) + 1$ only if $q_i = B$, and $y_{W-i} = 1$ or $q_i = C$, and $y_{W-i} = 1$. Thus, since $n_{OUT}(T_i) = n_i + 2$ by the induction hypothesis, we get that $n_{OUT}(T_{i+1}) = n_{i+1} + 2$. $\square$

D. Average Range Expansion for Extremal Ranges

We now use the DFA of Section IV-C to derive a closed-form formula for the average expansion of an extremal range $[0, y]$, where y is drawn uniformly at random from $[0, 2^W - 1]$.[2] This average is defined formally as follows:

$$\begin{aligned} G(W) &= \mathbb{E}_{y:0 \le y \le 2^W - 1}\left(opt\left([0, y]\right)\right) \\ &= \frac{1}{2^W} \cdot \sum_{y:0 \le y \le 2^W - 1} opt\left([0, y]\right). \end{aligned} \quad (4)$$

Theorem 3: The average extremal range expansion function $G(W)$ satisfies

$$\begin{aligned} G(W) &= \frac{4}{9} + \frac{W}{3} + \frac{4}{9} \cdot \left(\frac{1}{2}\right)^W && \text{if } W \text{ is odd} \\ G(W) &= \frac{4}{9} + \frac{W}{3} + \frac{5}{9} \cdot \left(\frac{1}{2}\right)^W && \text{if } W \text{ is even.} \end{aligned} \quad (5)$$

Proof: To calculate $G(W)$, we derive a Markov chain from the DFA of Section IV-C. This *Markov chain* is shown in Fig. 2(c). It has the same states as the DFA with the same interpretation. At each state, it flips a coin and takes the transition that corresponds to an input of 1 with probability 1/2, and the transition that corresponds to an input of 0 with probability 1/2. This simulates the DFA on an extremal range drawn uniformly at random.

The transition probabilities are represented in a 3×3 transition matrix P. The first row and column correspond to state A, the second to state B, and the third to state C. The (i, j)th element of P describes the transition probability from the state corresponding to row i to the state corresponding to column j

$$P = \begin{pmatrix} 0.5 & 0.5 & 0 \\ 0.5 & 0 & 0.5 \\ 0 & 0.5 & 0.5 \end{pmatrix}. \quad (6)$$

Let $r_i = (\Pr(q_i = A), \Pr(q_i = B), \Pr(q_i = C))$. Then, clearly $r_0 = (1, 0, 0)$, and by the properties of Markov chains, $r_i = r_0 \cdot P^i$.

By Theorem 2, $G(W)$ can be calculated based on the average number of transitions of the form $\delta(B, 1) = A$ or $\delta(C, 1) = B$ that the DFA performs on $y \in [0, 2^W - 1]$.

[2]In real-life classifiers, when there are ranges that appear more often than others, this theoretical analysis is not applicable.

Let n_y be a random variable that equals the number of these specific transitions while processing y. We present n_y as the sum of W indicator random variables $\{I_{y,i} | i \in [1,W]\}$, such that the function $I_{y,i}$ indicates whether the ith transition is one of the two specific transitions. Then, we can compute $G(W)$as follows:

$$\begin{aligned}
&G(W)\\
&\quad= \mathbb{E}_{y:0\le y\le 2^W-1}\left(opt\left([0,y]\right)\right)\\
&\quad= \mathbb{E}_{y:0\le y\le 2^W-1}(n_y+1) = 1+\mathbb{E}_{y:0\le y\le 2^W-1}(n_y)\\
&\quad= 1+\sum_{i=1}^{W}\mathbb{E}_{y:0\le y\le 2^W-1}(I_{y,i})\\
&\quad= 1+\sum_{i=1}^{W}\big(\Pr(q_{i-1}=B, y_{W-i+1}=1)\\
&\qquad+\Pr(q_{i-1}=C, y_{W-i+1}=1)\big)\\
&\quad= 1+\frac{1}{2}\cdot\sum_{i=0}^{W-1}\left(\Pr(q_i=B)+\Pr(q_i=C)\right)\\
&\quad= 1+\frac{1}{2}\cdot\sum_{i=0}^{W-1}\left(1-\Pr(q_i=A)\right) = 1+\frac{1}{2}\cdot\sum_{i=0}^{W-1}\left(1-r_i(1)\right)\\
&\quad= 1+\frac{1}{2}\cdot\sum_{i=0}^{W-1}\Big(1-\left((1,0,0)\cdot P^i\right)_{(1)}\Big)\\
&\quad= 1+\frac{1}{2}\cdot\sum_{i=0}^{W-1}\left(1-(P^i)_{(1,1)}\right). \qquad (7)
\end{aligned}$$

The matrix P satisfies $(P^{2i-1})_{(1,1)} = (P^{2i})_{(1,1)} = (1/3) + (2/3)\cdot(1/2)^{2i}$. Thus, the function $G(W)$ satisfies $G(W) = G(W-2)+(1/2)((1-(P^{W-2})_{(1,1)})+(1-(P^{W-1})_{(1,1)})) = G(W-2)+1-(P^{W-1})_{(1,1)} = G(W-2)+(2/3)-(4/3)\cdot(1/2)^W$ if W is odd and $G(W) = (1/2)(G(W-1)+G(W+1))$ if W is even. By solving these recurrence relations, we get the formula that appear in the theorem. □

To our knowledge, this is the *first formula in the literature* for the average encoding size of a nontrivial range set.

By [12], the worst-case expansion for an extremal range is $r^e(W) = r_p^e(W) = \lceil (W+1)/2\rceil$. Thus, clearly $G(W) \le \lceil (W+1)/2\rceil$. Theorem 3 and its corollary below show that the average encoding length is only about 2/3 of the worst case.

Corollary 4: The average extremal range expansion function $G(W)$ satisfies

$$\lim_{W\to\infty}\frac{G(W)}{W} = \frac{1}{3}. \qquad (8)$$

V. Analytical Tools for Range Expansion Lower Bounds

To prove lower bounds on the TCAM worst-case expansion, we first define the hull of a set of (binary) strings in the same way as in our previous work [12].

Definition 5 (Hull): The hull of n strings $\{a^1,\ldots,a^n\}$, where $a^i = a_1^i\ldots a_W^i$, is the smallest cuboid containing $a^1,\ldots,a^n$. We denote it by $H(a^1,\ldots,a^n)$. Formally

$$\begin{aligned}H(a^1,\ldots,a^n) = \big\{x = x_1\ldots x_W \in \{0,1\}^W \,|\\ \forall j\in[1,W], x_j \in \left\{a_j^1,\ldots,a_j^n\right\}\big\}. \qquad (9)\end{aligned}$$

The hull $H(a^1,\ldots,a^n)$ corresponds to the TCAM entry $s(H) = s_1\ldots s_n$, where $s_j = a_j^1$ if $a_j^1 = a_j^2 = \ldots = a_j^n$, and $s_j = *$ otherwise. The entry $s(H)$ is the entry with the minimal number of $*$'s that all the strings $a^1,\ldots,a^n$ match. Each string in the hull is matched by this TCAM entry, and vice versa. This is captured precisely in the following proposition, also from [12].

Proposition 5: Let $a^1,\ldots,a^n$ be n strings. Then, $a^1,\ldots,a^n$ match the same TCAM entry s if and only if all the strings in the hull $H(a^1,\ldots,a^n)$ match this TCAM entry.

We now want to introduce a novel general analytical tool that can help us analyze the minimum number of TCAM entries needed to encode a range. Intuitively, a *conflicting set of pairs* of size n is composed of n pairs of points. Each pair consists of one point in the range and one outside the range. We show that the pairs are *pairwise conflicting* such that we cannot encode together two points within the range or alternatively two points outside the range from two different pairs. We then deduce that these n pairs cannot be encoded with less than n TCAM entries. This analytical tool is first presented here and is stronger than previous tools designed for a similar purpose. It will enable us to obtain the improved lower bounds on the worst-case expansion presented later in Section VI. Recall that we work over the set of strings of length W, and the width of a TCAM entry is W.

Definition 6 (Conflicting Set of Pairs): A *conflicting set of pairs* B_n of size n with respect to a range R is defined as a set of n pairs of strings

$$B_n = \left\{(a^i,b^i) \mid i\in[1,n], \forall i\in[1,n], a^i,b^i\in\{0,1\}^W\right\}$$

that satisfies the following two conditions:

(i) *Alternation*: For $i\in[1,n]$

$$a^i \in R \quad\text{and}\quad b^i\notin R. \qquad (10)$$

(ii) *Hull*: For any i_1, i_2 such that $1\le i_1 < i_2 \le n$

$$\begin{aligned}H(a^{i_1},a^{i_2})\cap\{b^{i_1},b^{i_2}\}\neq\emptyset\\ H(b^{i_1},b^{i_2})\cap\{a^{i_1},a^{i_2}\}\neq\emptyset. \qquad (11)\end{aligned}$$

Since the alternation property holds for any pair and the hull property holds for any two pairs, we can easily observe the following.

Corollary 5: Let n be a positive integer, and $B_{n+1} = \{(a^1,b^1),\ldots,(a^{n+1},b^{n+1})\}$ be a conflicting set of pairs of size $n+1$. Then, for any $1\le i\le n+1$, $B_{n+1}\setminus\{(a_i,b_i)\}$ is a conflicting set of pairs of size n.

The next lemma gives a lower bound on the range expansion of a range with a conflicting set of pairs. Its proof can be found in [34].

Lemma 1: A range with a conflicting set of pairs of size n cannot be encoded in less than n TCAM entries.

VI. Bounds on Worst-Case Expansion

In this section, we study the exact worst-case expansions of one-dimensional and two-dimensional ranges. We present upper bounds on these expansions and then prove their tightness.

A. General 1-D Ranges

We start with finding the exact value of the worst-case expansion of one-dimensional ranges. To do so, we rely on an upper bound from [12]. We then suggest a simple encoding that achieves this known bound. Finally, we show the tightness of this bound based on the new analytical tool from Section V.

It is known [12] that the maximum range expansion $r(W)$ satisfies $r(W) \leq r_p(W) = W$. In this section, we describe a very simple algorithm that encodes a W-bit range with at most W rules. (This algorithm has the same maximum range expansion as a previously known encoding scheme [12], but it is much simpler.) This algorithm either uses *in* entries that encode the range, or *out* entries that encodes the complement of the range and an additional *in* entry that matches everything else. We compare the number of TCAM entries needed for each of the two alternatives and simply pick the alternative with the least TCAM entries.

We consider a W-bit range $R = [y, z] = \{y_1 \dots y_W, \dots, z_1 \dots z_W\}$. If $y = z$, then R is an exact match and can be encoded in one entry of the form $y \to \text{in}$. Otherwise, let $j \in [1, W]$ be the first bit index in which $y_1 \dots y_W$ and $z_1 \dots z_W$ differ, that is $y_1 \dots y_{j-1} = z_1 \dots z_{j-1}$, $y_j = 0$, and $z_j = 1$.

Let $n_0(y)$ and $n_1(y)$ be the number of 0's and the number of 1's in $y_{j+1} \dots y_W$, respectively. Similarly, let $n_0(z)$ and $n_1(z)$ be the number of 0's and the number of 1's in $z_{j+1} \dots z_W$, respectively. We can present R as a union of at most $n_0(y) + n_1(z) + 2$ prefix ranges by observing that

$$R = \left(\bigcup_{i \in [j+1, W], y_i = 0} \left\{ y_1 \dots y_{i-1} 1 (*)^{W-i} \right\} \right) \bigcup \left(\bigcup_{i \in [j+1, W], z_i = 1} \left\{ z_1 \dots z_{i-1} 0 (*)^{W-i} \right\} \right) \bigcup \{y, z\}. \quad (12)$$

A sequence of *in* TCAM entries each corresponding to a prefix in (12) encodes R in $n_0(y) + n_1(z) + 2$ prefix entries. Similarly, we can represent $R' = \{y_1 \dots y_{j-1} (*)^{W-j+1}\} \cap R^c$ as a union of $n_1(y) + n_0(z)$ prefix ranges

$$R' = \left(\bigcup_{i \in [j+1, W], y_i = 1} \left\{ y_1 \dots y_{i-1} 0 (*)^{W-i} \right\} \right) \bigcup \left(\bigcup_{i \in [j+1, W], z_i = 0} \left\{ z_1 \dots z_{i-1} 1 (*)^{W-i} \right\} \right). \quad (13)$$

It follows that we can also encode R by a sequence of *out* entries each corresponding to a prefix in (13) and the entry $\{y_1 \dots y_{j-1} (*)^{W-j+1}\} \to \text{in}$. This encoding is of size $n_1(y) + n_0(z) + 1$. We define $n_\oplus(R) = n_0(y) + n_1(z) + 2$ to be the number of entries in the first encoding and $n_\ominus(R) = n_1(y) + n_0(z) + 1$ to be the number of entries in the second encoding.

By definition, $n_0(y) + n_1(y) = W - j \leq W - 1$ and $n_0(z) + n_1(z) \leq W - 1$, so $n_\oplus(R) + n_\ominus(R) = (n_0(y) + n_1(z) + 2) + (n_1(y) + n_0(z) + 1) \leq 2(W - 1) + 3 = 2W + 1$. It follows that $\min\{n_\oplus(R), n_\ominus(R)\} \leq W$: the smaller of the two encodings includes at most W entries. We encode R by this encoding.

Example 4: Let $W = 5$, and consider the range $R = [y, z] = [5, 22] = \{00101, \dots, 10110\}$. As illustrated in Fig. 3(a), we can encode R as a union of $n_\oplus(R) = n_0(y) + n_1(z) + 2 = 2 + 2 + 2 = 6$ (here, $j = 1$) prefix TCAM entries $(01(*)^3 \to \text{in}, 0011* \to \text{in}, 100(*)^2 \to \text{in}, 1010* \to \text{in}, 00101 \to \text{in}, 10110 \to \text{in})$. We can also encode

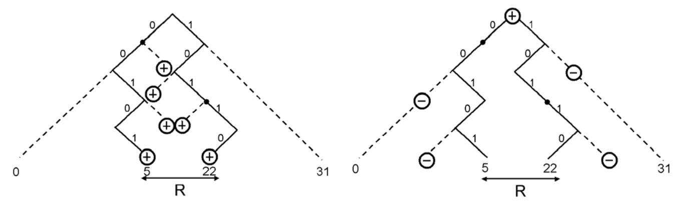


Fig. 3. Two encoding schemes of the range $R = [5, 22] = \{00101, \dots, 10110\}$ from Example 4. (a) Encoding of R itself as a union of several prefix ranges. The six plus signs represent the six TCAM entries in this encoding. (b) Alternative encoding of R in which we first negatively encode (with *out* entries) the complement of R and then add an additional *in* entry that matches R itself. Again, the five signs represent the five entries of this encoding. For any W-bit range R, one of these two encodings has at most W entries.

R, as presented in Fig. 3(b), by first encoding R^c by *out* entries followed by a last *in* entry that matches everything: $(000(*)^2 \to \text{out}, 00100 \to \text{out}, 11(*)^3 \to \text{out}, 10111 \to \text{out}, (*)^5 \to \text{in})$. This encoding consists of $n_\ominus(R) = n_1(y) + n_0(z) + 1 = 2 + 2 + 1 = 5$ entries. Here, we have that $n_\ominus(R) = 5 \leq W$.

The next theorem shows that the upper bound $r(W) \leq W$ on the maximum range expansion is actually tight. In [12], it is proved that the bound is tight for *prefix encoding schemes* (that is, $r_p(W) = W$). Here, we show that it is tight among *all TCAM encoding schemes* (that is, $r(W) = W$).

Theorem 6: For all $W \geq 1$, the maximum range expansion satisfies

$$r(W) = r_p(W) = W. \quad (14)$$

Proof: We show that for all $W \geq 1$, the maximum range expansion satisfies $r(W) \geq W$. Since $r(W) \leq r_p(W) \leq W$ (by [12] and the construction above), it then follows that $r(W) = r_p(W) = W$.

We first assume that W is even, and consider the range $R = [(1/3)(2^W - 1), 2^{W-1} - 1] \bigcup [2^{W-1}, (2/3)(2^W - 1)] = [(1/3)(2^W - 1), (2/3)(2^W - 1)] = \{(01)^{W/2}, \dots, (10)^{W/2}\}$. We show a conflicting set of pairs of size W for R. Then, by Lemma 1, we conclude that R cannot be encoded by less than W TCAM entries.

The construction is as follows. We start by defining $2W$ elements $c^1, c^2, \dots, c^{2W}$ from which we will assemble the W pairs of the conflicting set of pairs. We define $c^1 = (01)^{W/2}$, and for $i = 1, \dots, W$, we obtain c^{i+1} by flipping the $(W - (i-1))$th bit of c^i. That is, we get $c^2 = (01)^{(W/2)-1}00$ by flipping the least significant bit of c^1. Then, we get $c^3 = (01)^{(W/2)-1}10$ by flipping the $(W - 1)$th bit of c^2, and likewise until we obtain $c^W = 00(10)^{(W/2)-1}$ and $c^{W+1} = (10)^{W/2}$. We then continue to obtain $c^{W+2}, \dots, c^{2W}$ in a similar way, such that c^{i+1} is given by flipping the $(2W - (i-1))$th bit of c^i, for $i \in [W+1, 2W - 1]$.

We can see that $R = [(1/3)(2^W - 1), (2/3)(2^W - 1)] = \{(01)^{W/2}, \dots, (10)^{W/2}\} = [c^1, c^{W+1}]$. We recall that when comparing two W-bit binary strings c^i and c^j by the lexicographic order, $c^i < c^j$ iff there exists a bit k such that the first $k - 1$ bits of c^i and c^j are equal, the kth bit of c^i is 0, and the kth bit of c^j is 1. We now observe that $c^1, c^2, \dots, c^{2W}$ satisfy the following properties.

(i) For $i \in [1, W]$, the most significant bit of c^i is 0 and $c^i \in [0, 2^{W-1} - 1]$. Likewise, for $i \in [W+1, 2W]$, the most significant bit of c^i is 1 and $c^i \in [2^{W-1}, 2^W - 1]$.

(ii) For $i \in [1, W+1]$, the $W - (i-1)$ most significant bits of c^i are as of $c^1 = (01)^{W/2}$, and the $i-1$ least significant bits of c^i are as of $c^{W+1} = (10)^{W/2}$. For $i \in [W+1, 2W]$, the $2W - (i-1)$ most significant bits of c^i are as of c^{W+1}, and the $i - (W+1)$ least significant bits of c^i are as of $c^1 = (01)^{W/2}$.

(iii) For $i \in [2, W]$, the most significant bit in which c^i and c^1 differ is the $(W-(i-2))$th bit. Since $c^1_{W-(i-2)} = 0$ if i is odd and $c^1_{W-(i-2)} = 1$ if i is even, we have that $c^i \geq c^1$, $c^i \in R$ if i is odd, and $c^i < c^1$, $c^i \notin R$ if i is even. For the same reason, by comparing c^i and c^{W+1}, we also have that for $i \in [W+1, 2W]$, $c^i \in R$ if i is odd, and $c^i \notin R$ if i is even.

We now define for $i \in [1, W]$, $a^i = c^{2i-1}$, and $b^i = c^{2i}$. To show that $B_W = \{(a^1, b^1), \ldots, (a^W, b^W)\}$ is a conflicting set of pairs of size W, we have to show it satisfies the alteration property and the hull property. The alternation property follows directly from property (iii).

To show that B_W satisfies the hull property, consider two elements a^{i_1}, a^{i_2} for $1 \leq i_1 < i_2 \leq W$. If $i_1, i_2 \in [1, W/2]$ or $i_1, i_2 \in [(W/2)+1, W]$, then $b^{i_1} \in H(a^{i_1}, a^{i_2})$ since it shares $W-1$ of its W bits with a^{i_1} and the remaining bit with a^{i_2}. If $i_1 \in [1, W/2]$ and $i_2 \in [(W/2)+1, W]$, let $i = i_1$ and $j = i_2 - (W/2)$. We then have $a^{i_1} = a^i = (01)^{(W/2)-(i-1)}(10)^{(i-1)}$ and $a^{i_2} = a^{j+(W/2)} = (10)^{(W/2)-(j-1)}(01)^{(j-1)}$. We distinguish two possible subcases. If $i \geq j$, a^{i_1}, a^{i_2} differ in their $W - 2(i-1)$ most significant bits. Since a^{i_1} and b^{i_1} differ only in their $(W-2(i-1))$ most significant bit, we have that $b^{i_1} \in H(a^{i_1}, a^{i_2})$. For the same reason, if $i < j$, then $b^{i_2} \in H(a^{i_1}, a^{i_2})$.

We now consider two elements b^{i_1}, b^{i_2} such that $1 \leq i_1 < i_2 \leq W$. If $i_1, i_2 \in [1, W/2]$ or $i_1, i_2 \in [(W/2)+1, W]$, then $a^{i_2} \in H(b^{i_1}, b^{i_2})$ since a^{i_2}, b^{i_2} differ in a single bit on which a^{i_2} and b^{i_1} agree. If $i_1 \in [1, W/2]$ and $i_2 \in [(W/2)+1, W]$, we define i and j as above. Here, $b^{i_1} = b^i = (01)^{(W/2)-i}00(10)^{(i-1)}$ and $b^{i_2} = b^{j+(W/2)} = (10)^{(W/2)-j}11(01)^{(j-1)}$. If $i \geq j$, b^{i_1}, b^{i_2} differ (at least) in their last $2j-1$ bits with indices $\{W-(2j-2), \ldots, W\}$. Since a^{i_2}, b^{i_2} differ only in their $(W-(2j-2))$th bit, we have that $a^{i_2} \in H(b^{i_1}, b^{i_2})$. From the same reason, if $i < j$, then $a^{i_1} \in H(b^{i_1}, b^{i_2})$.

It follows that B_W is a conflicting set of pairs of size W, so by Lemma 1, R cannot be encoded in less than W TCAM entries.

If W is odd, the proof is analogous using the range $R = [(1/3)(2^{W-1} - 1), (4/3)(2^{W-1} - 1)] = \{0(01)^{(W-1)/2}, \ldots, (10)^{(W-1)/2}0\}$. □

B. General 2-D Ranges

We now find the exact maximum expansion of two-dimensional ranges. We show upper bounds on this expansion and then prove the tightness of the bounds by exposing a hard-to-encode two-dimensional range.

A two-dimensional range R^2 is defined as the product of two one-dimensional ranges $R_x \times R_y$, and the encoding of such a range should positively encode exactly the pairs of strings (a, b) such that $a \in R_x$ and $b \in R_y$.

We generalize the definition of $r(W)$ to multidimensional ranges, and define $r^d(W)$ as the maximum expansion of a

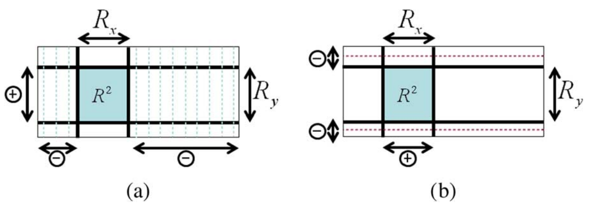


Fig. 4. Two-dimensional range $R^2 = R_x \times R_y$. (a) Encoding of the two-dimensional range R^2 by negatively (with *out* entries) encoding the complement of R_x and then positively (with *in* entries) encoding R_y. (b) Alternative encoding of R^2 by first negatively encoding the complement of R_y and then positively encoding R_x.

d-dimensional range in $[0, 2^W - 1]^d$. Likewise, define $r^{e,d}(W)$ as the maximum expansion of a d-dimensional *extremal* range, i.e., the maximum expansion of a range whose projection on each dimension is an extremal range. Finally, let $r^d_p(W)$ (respectively $r^{e,d}_p(W)$) be the maximum expansion of a (an extremal) d-dimensional range when we use only prefix encodings.

We begin with presenting an upper bound on the maximum expansion.

Lemma 2: The worst-case expansion $r^2(W)$ of a two-dimensional classification rule R^2 satisfies

$$r(W) \leq 2W. \tag{15}$$

Proof: The proof is based on the two possible encodings of a one-dimensional range presented at the beginning of Section VI. We also use similar notations. We consider a two-dimensional range $R^2 = R_x \times R_y$ and present two possible encodings of R^2 such that one of them has the required expansion.

First, as illustrated in Fig. 4(a), we encode R^2 starting with a sequence of *out* entries consisting of the Cartesian product of an encoding of the complement of R_x and $\{(*)^W\}$ (the second field in all entries is $(*)^W$). This sequence consists of $n_\ominus(R_x) - 1$ entries (here an additional entry that positively encodes R_x itself is not required). The encoding of R^2 continues with a sequence of *in* entries that are the product of an encoding of R_y itself with $\{(*)^W\}$. The length of this sequence is $n_\oplus(R_y)$. This encodes R^2 in $n_\ominus(R_x) - 1 + n_\oplus(R_y)$ entries.

We obtain the second encoding similarly, as demonstrated in Fig. 4(b). We start with a sequence of *out* entries whose projection on the y axis encode the complement of R_y followed by a sequence of *in* entries whose projections on the x-axis encode R_x. The length of this encoding is $n_\ominus(R_y) - 1 + n_\oplus(R_x)$. As explained earlier in this section, $n_\oplus(R_x) + n_\ominus(R_x)$ and $n_\oplus(R_y) + n_\ominus(R_y)$ are at most $2W+1$. It follows that $n_\ominus(R_x) - 1 + n_\oplus(R_y) + n_\ominus(R_y) - 1 + n_\oplus(R_x) = n_\oplus(R_x) + n_\ominus(R_x) + n_\oplus(R_y) + n_\ominus(R_y) - 2 \leq 4W$. Thus, $\min\{n_\ominus(R_x) - 1 + n_\oplus(R_y), n_\ominus(R_y) - 1 + n_\oplus(R_x)\} \leq 2W$ so the smaller of the two encodings has the desired expansion. □

We now show that our suggested encoding scheme for two-dimensional ranges has the optimal worst-case range expansion. To do so, we present a particular two-dimensional range, denoted by R^2, and show that we can build a conflicting set of pairs of size $2W$ for R^2. Then, from Lemma 1, we deduce that R^2 cannot be encoded in less than $2W$ TCAM entries.

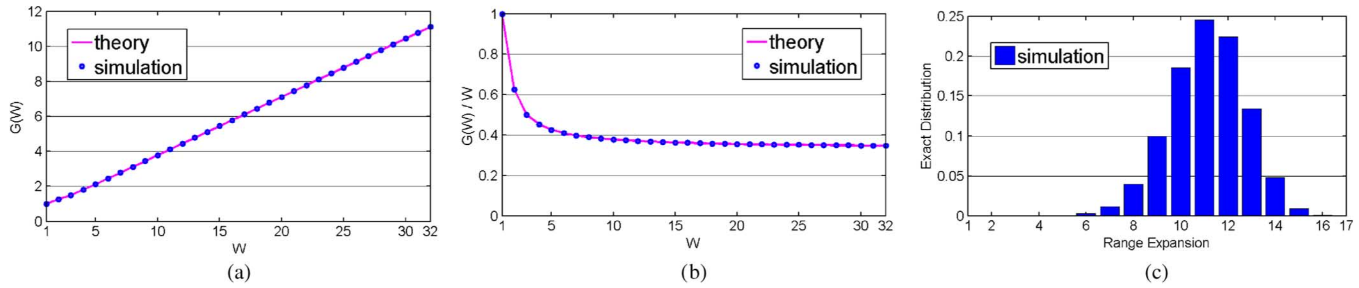


Fig. 5. Simulations of extremal range expansion. (a) Average extremal range expansion $G(W)$ presented in Theorem 5. (b) Normalized average extremal range expansion $G(W)/W$. We can see that indeed $\lim_{W\to\infty}(G(W)/W) = 1/3$ as stated in Corollary 4. (c) Extremal range expansion distribution for $W = 32$. The minimal expansion is 1, and the maximal expansion is $\lceil (W+1)/2 \rceil = 17$.

Lemma 3: The worst-case expansion of a two-dimensional classification rule R^2 satisfies

$$r(W) \geq 2W. \tag{16}$$

Proof Outline: The full proof is relatively straightforward but long and appears in [34]. We first generalize Definition 5 for pairs of strings and provide the proof for even values of W. We consider the range $R^2 = R \times R = [(1/3)(2^W - 1), (2/3)(2^W - 1)] \times [(1/3)(2^W - 1), (2/3)(2^W - 1)]$. The projection of R^2 on each dimension is the hard-to-encode range $R = [(1/3)(2^W - 1), (2/3)(2^W - 1)]$ from which we can build a conflicting set of pairs of size W, as described for the one-dimensional case. We also reuse the definitions of $a^1, b^1, \ldots, a^W, b^W$ defined there. We construct $4W$ pairs of strings as follows. For $i \in [1, W/2]$, $u^i = (a^i, a^1)$ and $v^i = (b^i, a^1)$. For $i \in [(W/2)+1, W]$, $u^i = (a^1, a^i)$ and $v^i = (a^1, b^i)$. Next, for $i \in [W+1, 3W/2]$, $u^i = (a^{i-(W/2)}, a^{(W/2)+1})$ and $v^i = (b^{i-(W/2)}, a^{(W/2)+1})$. Finally, for $i \in [(3W/2)+1, 2W]$, $u^i = (a^{(W/2)+1}, a^{i-(3W/2)})$ and $v^i = (a^{(W/2)+1}, b^{i-(3W/2)})$. Then, to obtain the result, we show that $\{(u^1, v^1), \ldots, (u^{2W}, v^{2W})\}$ is a conflicting set of pairs with $2W$ pairs of pairs of strings. □

The following theorem summarizes the result that follows from Lemmas 2 and 3.

Theorem 7: The worst-case expansion of a two-dimensional classification rule satisfies

$$r(W) = rp(W) = 2W. \tag{17}$$

Proof: Clearly, $r^2(W) \leq r_p^2(W)$. Since the encoding presented in the proof of Lemma 2 includes only prefix entries, we have also that $r_p^2(W) \leq 2W$. Finally, by Lemma 3, we have the result. □

C. Extremal 2-D Ranges

By Theorem 6 and the mentioned result of [12] regarding the exact value of the worst-case expansion of one-dimensional extremal ranges, the worst-case expansion is improved when only extremal ranges are considered. The next theorem shows that a similar improvement exists also for two-dimensional ranges.

Theorem 8: The worst-case expansion of a two-dimensional extremal classification rule satisfies

$$2 \cdot \left\lceil \frac{W+1}{2} \right\rceil - 1 \leq r^{e,2}(W) \leq r_p^{e,2}(W) \leq W+1. \tag{18}$$

More specifically, if W is even, $r^{e,2}(W) = r_p^{e,2}(W) = W+1$ and if W is odd $W \leq r^{e,2}(W) \leq r_p^{e,2}(W) \leq W+1$.

Proof Outline: See [34] for the full proof. We consider the same two possible encodings suggested earlier for general two-dimensional ranges. We prove that for two-dimensional extremal ranges, the smaller of the two achieves the improved upper bound. Later, we present a range with a conflicting set of pairs of the required size to get the lower bound. □

VII. Experimental Results

A. One-Dimensional Extremal Ranges

We performed simulations to verify the results of the average range expansion for extremal ranges presented in Section IV-D. Fig. 5(a) presents the function $G(W)$ for $W \in [1, 32]$. For each value of W, we averaged all 2^W extremal ranges of the form $[0, y]$. We can see that the simulated average expansion exactly matches the theory from Theorem 3. For instance, $G(W = 3) = 1.5$ since the ranges [0, 0], [0, 1], [0, 3], [0, 7] can be encoded in one TCAM entry while the encodings of the ranges [0, 2], [0, 4], [0, 5], [0, 6] requires two entries.

Fig. 5(b) presents the function $G(W)/W$ for $W \in [1, 32]$. We can see that indeed $\lim_{W\to\infty}(G(W)/W) = 1/3$ as stated in Corollary 4. For instance, for $W = 16$, $G(W)/W \approx 0.3611$, and for $W = 32$, $G(W)/W \approx 0.3472$.

Lastly, Fig. 5(c) presents the distribution of the extremal range expansion for $W = 32$. The minimal expansion is of course 1, and the maximal expansion is $\lceil (W+1)/2 \rceil = 17$, both with negligible probability.

B. Two-Dimensional Ranges

We would like to examine the average expansion of two-dimensional ranges in $[0, 2^W - 1] \times [0, 2^W - 1]$. We consider the encoding scheme for two-dimensional ranges described in Section VI (with an improved worst-case expansion of $2W$) in comparison to other well-known encoding schemes such as the Binary Prefix encoding [11], the SRGE encoding [9], and the external encoding for two-dimensional ranges from [12].

Table II summarizes the results. The improvement in the average expansion is more significant for larger values of W. For instance, for $W = 8$, the average expansion of the suggested scheme is 4.85 in comparison to 36.56, 26.42, and 12.98 in the first three schemes, an improvement of 86.7%, 81.6%, and 62.6%, respectively.

C. Real-Life Database Statistics

We examine the frequency of generalized extremal rules in a real-life database of 120 separate rule files with 214 941 rules originating from various applications (such as firewalls, and ACL in routers). The same database was also used in [1], [9], and [10]. The rules in this database are defined on the typical five

TABLE II
RANGE EXPANSION FOR TWO-DIMENSIONAL RANGES IN $[0, 2^W - 1] \times [0, 2^W - 1]$

Encoding Scheme	Worst-Case Expansion	Average Expansion				
		$W=4$	$W=5$	$W=6$	$W=7$	$W=8$
Binary Prefix	$(2W-2)^2$	6.14	10.72	17.26	25.86	36.56
SRGE	$(2W-4)^2$	4.03	6.96	11.51	17.95	26.42
External Encoding	$4W-3$	5.24	7.06	9.00	10.98	12.98
Suggested Scheme	$2W$	1.84	2.45	3.18	3.99	4.85

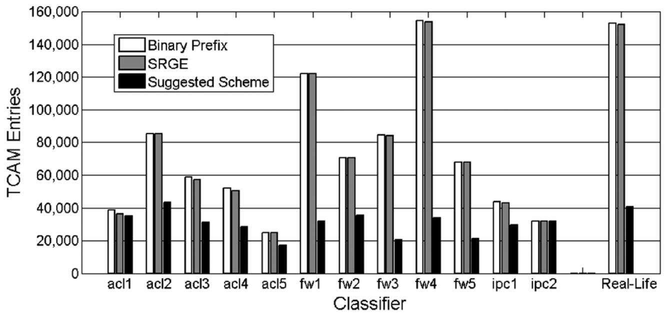
fields and follow the description in the introduction. Ranges can appear only in the source port or in the destination port, while the requirement for the other fields is either a prefix or an exact match that can be encoded without any expansion. The source port and the destination port are W-bit fields (with $W = 16$). We find that out of the 214 941 rules, *97.2% (208 850) are generalized extremal rules*, i.e., all their fields contain generalized extremal ranges. Even when excluding the exact-match rules, *89.4% of the remaining rules are still generalized extremal rules* (51 065 rules out of 57 146).

D. Effectiveness on Real-Life Packet Classifiers

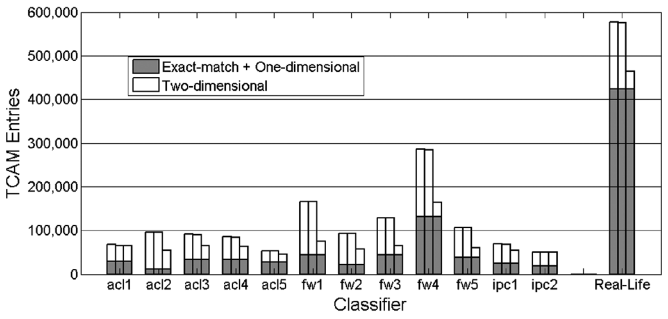
We now consider 12 artificial classifiers generated by the ClassBench benchmark tool [35] in addition to the union of the 120 real-life rule files from Section VII-C. These 12 artificial files are of three families: access control lists (files acl1-acl5), firewalls (files fw1-fw5), and IP chains (files ipc1-ipc2). To produce them, we use the original 12 parameter files of the tool, as in [9]. The number of rules in each file was in the range [40 362, 50 000]. Also in these files, ranges appear only in the two port fields.

Fig. 6(a) presents the results. We compared the expansion of our encoding scheme for two-dimensional ranges (with the upper bound of $2W$) versus Binary Prefix encoding [11] and SRGE encoding [9]. For the classifier *fw4*, for instance, the total expansion using our scheme is 33 774 entries in comparison to 154 813 and 153 691 entries using Binary Prefix encoding and SRGE encoding, respectively—an improvement of 78.2% and 78.0%, respectively. Furthermore, the results for the different files in each of the three families were similar. This can be explained by the fact that files of the same family have a similar (although non identical) distribution of complicated ranges that have large expansions. Likewise, for the real-life files, the improvement is 73.4% in comparison to Binary Prefix.

Fig. 6(b) compares the total expansion of all rules in these classifiers in the regular TCAM architecture using Binary Prefix encoding and SRGE encoding (illustrated in the two left bars in each group of three) and in our In/Out TCAM architecture from Fig. 1 (in the right bar). In this simulation, we choose to encode all the two-dimensional ranges in the modified TCAM of the new architecture using *in* and *out* entries in order to improve their average expansion. Therefore, the expansion of exact-match rules and one-dimensional rules (encoded in the first part of the architecture with only *in* entries) is exactly as in Binary Prefix encoding, and the total improvement is less significant but still not negligible. For instance, for the real-life files, the improvement in the total expansion is 19.5% with respect to Binary Prefix encoding. This essentially serves as a proof of concept to our In/Out TCAM architecture.



(a)



(b)

Fig. 6. Effectiveness of our encoding scheme and the suggested In/Out TCAM architecture (illustrated in Fig. 1) on 12 artificial classifiers generated by ClassBench benchmark tool and on a real-life database. For each classifier, the two left bars present the expansion of Binary Prefix encoding and of SRGE encoding, while the third bar illustrates the expansion of our suggested solution. (a) We compare the total expansion of the two-dimensional ranges of the classifiers. (b) We examine the expansion using the In/Out TCAM architecture when the two-dimensional ranges are encoded in the modified TCAM, i.e., the white bars correspond to (a).

VIII. CONCLUSION

In this paper, we presented a novel combined TCAM architecture, composed of a regular TCAM and a modified TCAM, which enables independent encoding of each rule in a set of rules, providing a guaranteed improved expansion at the cost of additional logic. Motivated by this architecture, we studied how to optimally encode a single range rule. We presented an encoding algorithm that is optimal for all possible *generalized extremal rules*, which represent 89% of all nontrivial rules in a typical real-life classification database. We also obtained tight bounds on the worst-case expansion for general classification rules, both for one-dimensional and two-dimensional ranges.

ACKNOWLEDGMENT

The authors would like to thank N. Nisan, D. Raz, A. Shpiner, and A. Bergman for their helpful suggestions.

REFERENCES

[1] D. E. Taylor, "Survey and taxonomy of packet classification techniques," *Comput. Surveys*, vol. 37, no. 3, pp. 238–275, 2005.
[2] G. Varghese, *Network Algorithmics*. San Mateo, CA, USA: Morgan Kaufmann, 2004.
[3] J. Chao and B. Liu, *High Performance Switches and Routers*. Hoboken, NJ, USA: Wiley, 2007.
[4] J. Naous, D. Erickson, G. A. Covington, G. Appenzeller, and N. McKeown, "Implementing an OpenFlow switch on the NetFPGA platform," in *Proc. ACM/IEEE ANCS*, 2008, pp. 1–9.
[5] NetLogic Microsystems, "NetLogic Microsystems,".
[6] Renesas, Tokyo, Japan, "Renesas," [Online]. Available: http://www.renesas.com/
[7] P. Gupta and N. McKeown, "Packet classification on multiple fields," in *Proc. ACM SIGCOMM*, 1999, pp. 147–160.

[8] S. Singh, F. Baboescu, G. Varghese, and J. Wang, “Packet classification using multidimensional cutting,” in *Proc. ACM SIGCOMM*, 2003, pp. 213–224.
[9] A. Bremler-Barr and D. Hendler, “Space-efficient TCAM-based classification using gray coding,” *IEEE Trans. Comput.*, vol. 61, no. 1, pp. 18–30, Jan. 2012.
[10] K. Lakshminarayanan, A. Rangarajan, and S. Venkatachary, “Algorithms for advanced packet classification with ternary CAMs,” in *Proc. ACM SIGCOMM*, 2005, pp. 193–204.
[11] V. Srinivasan, G. Varghese, S. Suri, and M. Waldvogel, “Fast and scalable layer four switching,” in *Proc. ACM SIGCOMM*, 1998, pp. 191–202.
[12] O. Rottenstreich, R. Cohen, D. Raz, and I. Keslassy, “Exact worst-case TCAM rule expansion,” *IEEE Trans. Comput.*, vol. 62, no. 6, pp. 1127–1140, Jun. 2013.
[13] S. Suri, T. Sandholm, and P. R. Warkhede, “Compressing two-dimensional routing tables,” *Algorithmica*, vol. 35, no. 4, pp. 287–300, 2003.
[14] R. Draves, C. King, S. Venkatachary, and B. D. Zill, “Constructing optimal IP routing tables,” in *Proc. IEEE INFOCOM*, 1999, pp. 88–97.
[15] T. Sasao, “On the complexity of classification functions,” in *Proc. ISMVL*, 2008, pp. 57–63.
[16] Y.-K. Chang, C.-I. Lee, and C.-C. Su, “Multi-field range encoding for packet classification in TCAM,” in *IEEE INFOCOM Mini-Conf.*, 2011, pp. 196–200.
[17] E. Spitznagel, D. E. Taylor, and J. S. Turner, “Packet classification using extended TCAMs,” in *Proc. IEEE ICNP*, 2003, pp. 120–131.
[18] H. Che, Z. Wang, K. Zheng, and B. Liu, “DRES: Dynamic range encoding scheme for TCAM coprocessors,” *IEEE Trans. Comput.*, vol. 57, no. 7, pp. 902–915, Jul. 2008.
[19] A. X. Liu, C. R. Meiners, and Y. Zhou, “All-match based complete redundancy removal for packet classifiers in TCAMs,” in *Proc. IEEE INFOCOM*, 2008, pp. 574–582.
[20] C. R. Meiners, A. X. Liu, and E. Torng, “Bit weaving: A non-prefix approach to compressing packet classifiers in TCAMs,” *IEEE/ACM Trans. Netw.*, vol. 20, no. 2, pp. 488–500, Apr. 2012.
[21] D. Applegate *et al.*, “Compressing rectilinear pictures and minimizing access control lists,” in *Proc. SODA*, 2007, pp. 1066–1075.
[22] A. X. Liu, C. R. Meiners, and E. Torng, “TCAM Razor: A systematic approach towards minimizing packet classifiers in TCAMs,” *IEEE/ACM Trans. Netw.*, vol. 18, no. 2, pp. 490–500, Apr. 2010.
[23] C. R. Meiners, A. X. Liu, E. Torng, and J. Patel, “Split: Optimizing space, power, and throughput for TCAM-based classification,” in *Proc. ACM/IEEE ANCS*, 2011, pp. 200–210.
[24] C. R. Meiners, A. X. Liu, and E. Torng, “Topological transformation approaches to TCAM-based packet classification,” *IEEE/ACM Trans. Netw.*, vol. 19, no. 1, pp. 237–250, Feb. 2011.
[25] O. Rottenstreich and I. Keslassy, “On the code length of TCAM coding schemes,” in *Proc. IEEE ISIT*, 2010, pp. 1908–1912.
[26] B. Schieber, D. Geist, and A. Zaks, “Computing the minimum DNF representation of Boolean functions defined by intervals,” *Discrete Appl. Math.*, vol. 149, no. 1–3, pp. 154–173, 2005.
[27] E. Norige, A. X. Liu, and E. Torng, “A ternary unification framework for optimizing TCAM-based packet classification systems,” in *Proc. ACM/IEEE ANCS*, 2013, pp. 95–104.
[28] A. X. Liu and M. G. Gouda, “Complete redundancy removal for packet classifiers in TCAMs,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 4, pp. 424–437, Apr. 2010.
[29] K. Kogan *et al.*, “SAX-PAC (Scalable and eXpressive PAcket Classification),” in *Proc. ACM SIGCOMM*, 2014, pp. 15–26.
[30] O. Rottenstreich, A. Berman, Y. Cassuto, and I. Keslassy, “Compression for fixed-width memories,” in *Proc. IEEE ISIT*, 2013, pp. 2379–2383.
[31] G. Rétvári, J. Tapolcai, A. Kőrösi, A. Majdán, and Z. Heszberger, “Compressing IP forwarding tables: Towards entropy bounds and beyond,” in *Proc. ACM SIGCOMM*, 2013, pp. 111–122.
[32] O. Rottenstreich *et al.*, “Compressing forwarding tables for datacenter scalability,” *IEEE J. Sel. Areas Commun.*, vol. 32, no. 1, pp. 138–151, Jan. 2014.
[33] W. Jiang and V. K. Prasanna, “Scalable packet classification on FPGA,” *IEEE Trans. VLSI Syst.*, vol. 20, no. 9, pp. 1668–1680, Sep. 2012.
[34] O. Rottenstreich *et al.*, “Optimal In/Out TCAM encodings of ranges,” Technion, Haifa, Israel, Tech. Rep., 2014 [Online]. Available: http://webee.technion.ac.il/people/or/publications/Optimal_TCAM_TR.pdf
[35] D. E. Taylor and J. S. Turner, “ClassBench: A packet classification benchmark,” in *Proc. IEEE INFOCOM*, 2005, vol. 3, pp. 2068–2079.

Ori Rottenstreich received the B.S. degree (*summa cum laude*) in computer engineering and Ph.D. degree in electrical engineering from the Technion, Haifa, Israel, in 2008 and 2014, respectively.

He is a Postdoctoral Research Associate with Princeton University, Princeton, NJ, USA. His current research interests include exploring novel coding techniques for networking applications.

Dr. Rottenstreich is a recipient of the Google Europe Fellowship in Computer Networking, the Andrew Viterbi Graduate Fellowship, the Jacobs-Qualcomm Fellowship, the Intel Graduate Fellowship, and the Gutwirth Memorial Fellowship. He also received the Best Paper Runner-Up Award at the IEEE INFOCOM 2013.

Isaac Keslassy (M'02–SM'11) received the M.S. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, USA, in 2000 and 2004, respectively.

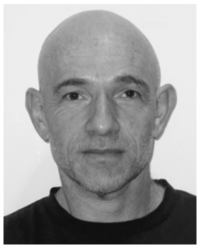
He is currently an Associate Professor with the Electrical Engineering Department, Technion, Haifa, Israel. His recent research interests include the design and analysis of high-performance routers and multicore architectures.

Dr. Keslassy has been the recipient of the ERC Starting Grant and the Alon, Mani, Yanai, and Taub awards.

Avinatan Hassidim received the doctoral degree in computer science from the Hebrew University, Jerusalem, Israel, in 2009.

He was later a Postdoctoral Fellow with the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, and is now a Senior Lecturer with the Department of Computer Science, Bar Ilan University, Ramat Gan, Israel. His primary research interest is in algorithms, focusing on distributed solutions and computer networks.

Dr. Hassidim received the Best Paper Award at SIGMETRICS 2011 and was the runner up for the Best Paper Award ate IEEE INFOCOM 2012 and 2013.



Haim Kaplan received the Ph.D. degree in computer science from Princeton University, Princeton, NJ, USA, in 1997.

He was a member of Technical Staff with AT&T Research from 1996 to 1999. Since 1999, he has been a Professor with the School of Computer Science, Tel Aviv University, Tel Aviv, Israel. His research interests are design and analysis of algorithms and data structures.

Ely Porat received the Ph.D. degree in computer science from Bar Ilan University, Ramat Gan, Israel, in 2001.

Thereafter, in parallel to a military service, he was an Assistant Professor with Bar Ilan University and became an Associate Professor. He also holds a Visiting Professor position with the University of Michigan, Ann Arbor, MI, USA, and Tel Aviv University, Tel Aviv, Israel, and he is currently visiting the Simon Institute for Theory of Computing, University of California, Berkeley, CA, USA. His primary research field is pattern matching, but he has also worked on streaming algorithms, data structures, coding theory, group testing, compressed sensing, distributed algorithms, and game theory. A big part of his work is interdisciplinary, and he has combined methods from different areas on more than one occasion.