On Finding an Optimal TCAM Encoding Scheme for Packet Classification

Ori Rottenstreich, Isaac Keslassy
Technion
{or@tx, isaac@ee}.technion.ac.il

Avinatan Hassidim
Google Israel & Bar-Ilan University
avinatanh@gmail.com

Haim Kaplan
Tel Aviv University
haimk@post.tau.ac.il

Ely Porat
Bar-Ilan University
porately@cs.biu.ac.il

***Abstract*—Hardware-based packet classification has become an essential component in many networking devices. It often relies on TCAMs (ternary content-addressable memories), which need to compare the packet header against a set of rules. But efficiently encoding these rules is not an easy task. In particular, the most complicated rules are range rules, which usually require multiple TCAM entries to encode them. However, little is known on the optimal encoding of such non-trivial rules.**

In this work, we take steps towards finding an optimal encoding scheme for *every* possible range rule. We first present an optimal encoding for all possible generalized extremal rules. Such rules represent 89% of all non-trivial rules in a typical real-life classification database. We also suggest a new method of simply calculating the optimal expansion of an extremal range, and present a closed-form formula of the average optimal expansion over all extremal ranges. Next, we present new bounds on the worst-case expansion of general classification rules, both in one-dimensional and two-dimensional ranges. Last, we introduce a new TCAM architecture that can leverage these results by providing a guaranteed expansion on the tough rules, while dealing with simpler rules using a regular TCAM. We conclude by verifying our theoretical results in experiments with synthetic and real-life classification databases.

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), then returns the first rule that matches the header fields, and applies a corresponding action on 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 *TCAMs* (ternary content-addressable associative memories) 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 0s, 1s, 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 are in specific intervals. Encoding range rules requires several TCAM entries (this is called *range expansion*), and therefore although most rules are simple rules, most entries are used to encode range rules [9]. In addition, there is evidence that the percentage of range-based rules is increasing [10].

TCAM devices have a large set of rules which can be encoded together. However, understanding how to do this efficiently (using as few TCAM entries as possible) requires us first to understand how a single rule can be encoded. Therefore, a large body of work has been devoted to this question. Still, *no practical algorithm can give an optimal encoding for any arbitrary range rule*, i.e. the encoding that minimizes the needed number of TCAM entries. Instead, because of the high complexity of devising such an optimal algorithm, past works have limited themselves to sub-optimal encoding. They have either restricted the degrees of freedom of the algorithm, e.g. by forcing it to be prefix [11] or only with *accept* entries [12], or employed heuristic approaches [1]–[3], [13]–[15].

B. Our Contributions

In this paper we study the fundamental complexity of encoding a single rule, using both *accept* and *deny* entries. We focus on *the optimal encoding of any single range*, and explore the theoretical hardness of encoding one-dimensional and two-dimensional ranges. Then, we deduce practical results for providing *guaranteed-expansion* TCAM encoding algorithms.

As illustrated in Table I and Fig. 1, we mainly provide *three new contributions* in this paper.

We first provide a theoretical contribution. Consider a set $\{0, 1, \cdots, 2^W - 1\}$ of 2^W points, also represented as a tree with 2^W leaves. *Extremal ranges* in this set 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 a subtree of this tree. For instance, within a tree with 64 leaves, $[5, 7]$ is extremal for the subtree that represents $[4, 7]$, and therefore is a generalized extremal range.

The first contribution of this paper is that *we achieve an optimal-encoding goal for generalized extremal ranges*. Specifically, we present 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 the proof that there is an optimal TCAM encoding for generalized extremal ranges that uses only prefix TCAM entries. In other words, we prove that for generalized extremal ranges, restraining

978-1-4673-5946-7/13/$31.00 ©2013 IEEE

TABLE I

SUMMARY OF THE NOVEL PAPER RESULTS (IN BOXED BOLD). (A) PRESENTS IN THE LAST ROW AN OPTIMAL ALGORITHM FOR THE ENCODING OF ONE-DIMENSIONAL (GENERALIZED) EXTREMAL RULES (WITH RANGES LIKE $[0, y]$). AS SHOWN, THIS IS THE FIRST OPTIMAL RESULT WHEN THE DEGREES OF FREEDOM OF THE ALGORITHM ARE NOT LIMITED. PREVIOUS PAPERS EITHER ASSUMED ADDITIONAL CONSTRAINTS ON THEIR ALGORITHMS OR DID NOT PROVIDE OPTIMAL ALGORITHMS.(B) PRESENTS THE NEW RESULTS OBTAINED BY THIS PAPER FOR GENERAL RANGES. AS SHOWN IN THE LAST ROW, WHEN THE DEGREES OF FREEDOM OF THE ALGORITHM ARE NOT LIMITED, THE PAPER ACHIEVES A TIGHT BOUND OF W FOR ONE-DIMENSIONAL GENERAL RANGES. IN ADDITION, IN TWO-DIMENSIONAL RANGES, THE PAPER OBTAINS TIGHT BOUNDS OF $W + 1$ FOR EXTREMAL RULES (ASSUMING EVEN W FOR SIMPLICITY), AS WELL AS $2W$ FOR GENERAL RULES.

(A) Optimal algorithm for any range

Constraints			References	Extremal Ranges		General Ranges	
No *deny* entries	Prefix code	Gray codes		One Dimension	Two Dimensions	One Dimension	Two Dimensions
x	x	-	[16]	$\surd$	-	$\surd$	-
x	-	x		-	-	-	-
x	-	-		-	-	-	-
-	x	-	[11]	$\surd$	$\surd$	$\surd$	$\surd$
-	-	-		$\boxed{\surd}$	-	-	-

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

Constraints			References	Extremal Ranges				General Ranges			
No *deny* entries	Prefix code	Gray codes		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	-	[16]	W	W	W^2	-	$2W-2$	$2W-2$	$(2W-2)^2$	-
x	-	x	[9]	-	-	-	-	$2W-4$	W	$(2W-4)^2$	-
x	-	-	[12]	W	-	W^2	-	$2W-4$	$2W-4$	$(2W-4)^2$	-
-	x	-	[11], [17]	$\lceil \frac{W+1}{2} \rceil$	$\lceil \frac{W+1}{2} \rceil$	$\boxed{\mathbf{W+1}}$	$\boxed{\mathbf{W+1}}$	W	W	$\boxed{\mathbf{2W}}$	$\boxed{\mathbf{2W}}$
-	-	-	[17]	$\lceil \frac{W+1}{2} \rceil$	$\lceil \frac{W+1}{2} \rceil$	$\boxed{\mathbf{W+1}}$	$\boxed{\mathbf{W+1}}$	W	$\boxed{\mathbf{W}}$	$\boxed{\mathbf{2W}}$	$\boxed{\mathbf{2W}}$

the degrees of freedom of our encoding does not affect its optimality. We can then use a simple dynamic programming algorithm to find the smallest TCAM that uses only prefix TCAM rules, based on an existing algorithm [11].

As shown in the last row of Table I(A), for the first time, we achieve optimal encoding for non-trivial ranges. Former results had to limit the degrees of freedom of the algorithms, e.g. by constraining the encoding to be prefix, and therefore achieved sub-optimal encoding.

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 consider a union of 120 real-life classification databases from [10]. It contains 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* (51,055 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. To our knowledge, this is the *first formula known in the literature* for the average encoding size of a non-trivial range set. We show that it is only $2/3$ of the worst case.

Our second main contribution is that *we find the exact worst-case expansions of the one-dimensional as well as the two-dimensional ranges* (last rows of Table I(B)). In particular, we first present a simple algorithm that is optimal for general one-dimensional ranges in the worst case. Specifically, we first show that it encodes any range in $[0, 2^W - 1]$ in at most W entries and exhibit a range that cannot be encoded with less than W TCAM entries. Thus, our algorithm is *optimal* for the worst case, and it improves substantially on a best previously-known lower-bound of $\lceil (W+1)/2 \rceil$ [17]. In addition, 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, we present an algorithm that encodes any two-dimensional range with at most $2W$ entries and present again a matching lower bound, i.e. a tough two-dimensional range such that any TCAM encoding it will necessarily contain at least $2W$ entries. Therefore, our presented algorithm is *optimal* again in the worst case.

Finally, our third and last contribution is a new *joint TCAM architecture*. As Fig. 1 illustrates, the architecture combines a regular TCAM together with a modified TCAM, which can provide a *guaranteed improved expansion* for the tough classification rules. More specifically, given a set of classification rules, we split the rules between the two TCAMs. We set the first TCAM to encode the simple rules, while the second TCAM encodes the complex ones (e.g., the two-dimensional or the non-extremal one-dimensional rules). Then, each incoming header is sent to both TCAMs. The first TCAM outputs the index of its first matching entry. The second TCAM provides the

index of the first matching entry to its simplified PE (priority encoder), which corresponds to the index of its first matching rule. To do so, it decomposes its PE logic into a hierarchical structure, in which the first module determines whether there is a match on each rule, then the second one determines the first matching rule. Then, a modified conversion module considers both indices. From each index, it can deduce the respective rule index in the original classification list. Thus, it knows which rule has higher priority, and can output the corresponding action. It is therefore slightly more complex than a regular TCAM conversion module, which can be implemented using a simple SRAM with the action for each index. Here, the modified conversion module needs to use two SRAM-based memory tables, where each table entry may contain the original rule index and the associated action corresponding to each index.

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 SDN (Software-Defined Networking) [10], [18]. 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. In addition, updates are easier to implement in the modified TCAM architecture [17], and therefore in the combined architecture. On the other hand, note that the modified TCAM involves additional logic, and is not an off-the-shelf component. Therefore, it is more costly, which may limit the current appeal of the new architecture.

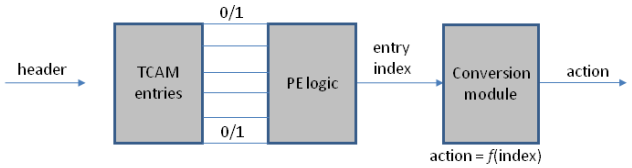
We further simulate the new architecture on the set of 120 real-life rule files mentioned above. We encode two-dimensional rules in the modified TCAM using the schemes in our paper, and other rules in the regular one using a simple binary prefix scheme. We find that the number of entries needed to encode the two-dimensional rules decreases by 73.4%, and the total number of entries needed decreases by 19.5%.

Due to space limits, some proofs are fully presented in [19].

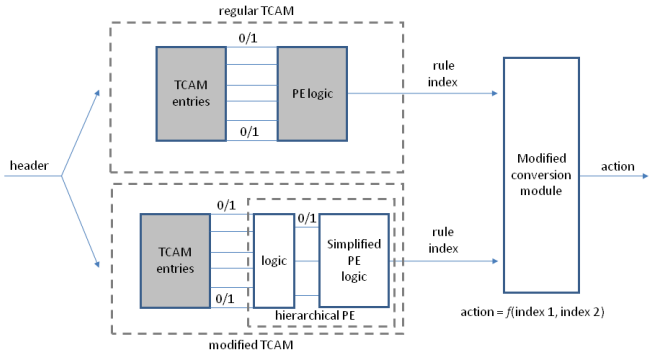
C. Related Work

As further illustrated in Table I, several previous papers have tried to find bounds on the worst-case expansion of a single rule. It is well-known that each range defined over a field of W bits can be encoded in at most $2W-2$ prefix TCAM entries for $W \geq 2$, where all TCAM entries have an action of *accept* [16]. For example, assume that $W=4$, and that we want to encode the single range $R=[1,14] \subseteq [0, 2^W-1]$ so that packets in that range are accepted while others are denied (default action). Then we need the following $2W-2=6$ TCAM entries, not counting the last default entry: $(0001 \rightarrow \text{accept}, 001* \rightarrow \text{accept}, 01** \rightarrow \text{accept}, 10** \rightarrow \text{accept}, 110* \rightarrow \text{accept}, 1110 \rightarrow \text{accept}, (****\rightarrow \text{deny}))$.

Using a non-prefix TCAM encoding, the upper bound of $2W-2$ was improved to $2W-4$ [20]. To show that, the disjunctive normal form representation of a range function was studied. In addition, Gray codes can be used instead of binary codes to reduce the worst-case TCAM size for any range from $2W-2$ to $2W-4$ as well [9].



(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 joint 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.

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.

The previous examples only used entries with the action of *accept*. In general, we also allow entries with the action of *deny*. In this case, the order of the TCAM entries becomes significant and the first entry that applies to a given input determines the action on that input. When both actions are allowed, there is an upper bound of W entries [17]. For instance, the range $R=[1,14]$ could be encoded using $3 \leq W$ entries: (0000$\rightarrow$ deny, 1111 $\rightarrow$ deny, **** $\rightarrow$ accept).

Other than the papers mentioned above, there is an extensive literature on *efficient heuristics* of how to encode ranges in TCAMs [1]–[3], [13]–[15], [21]–[24].

Some rules specify a range both for the source IP's and for the destination IP's. 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 accepting 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 constrained to the use of only accepting entries, then there is a range for which the encoding length has to contain at least W entries [9]. Furthermore, for binary codes, it was shown in [12] that there is a range whose encoding requires at least $2W-4$ accepting TCAM entries.

When both denying and accepting entries are used, [17] presented a lower bound of $\lceil \frac{W+1}{2} \rceil$ for extremal ranges given in binary codes, even when the entries are not limited to be prefix. For general ranges, a lower bound of W was suggested *only when the entries are limited to be prefix.*

Finally, an algorithm for finding an optimal prefix encoding for a given range is presented in [11]. However, its optimality is limited to encodings that contain only prefix entries.

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.

Definition 1 (Range, prefix range, extremal 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 say that the range is a* general range *when we want to emphasize that it is not necessarily a prefix range. When $r_1 = 0$ or $r_2 = 2^W - 1$ we call the range an* extremal range*.*

Definition 2 (TCAM entry, prefix TCAM entry)**.** *A* TCAM entry *S of width W is a ternary string $S = s_1 \dots s_W \in \{0,1,*\}^W$, where $\{0,1\}$ are bit values and $*$ stands for* don't-care*. A W-bit string $b = b_1 \dots b_W$ matches S, denoted as $b \in S$, if and only if for all $i \in [1, W]$, $s_i \in \{b_i, *\}$. We will use S to denote also the set of strings that it matches, when no confusion will arise.*

A TCAM entry $S = s_1 \dots s_W \in \{0,1,*\}^W$ 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 action a that is either *accept* or *deny*. We denote a pair consisting of an entry S and an action a by $S \to a$. Depending on the context, we shall refer by a *TCAM entry* either to S or to the pair $S \to a$.

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 its action is to accept all 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 3 (TCAM Encoding of a range)**.** *A* TCAM encoding *ϕ of a range R of width W is a set of TCAM entries $(S_1 \to a_1, \dots, S_n \to a_n)$ where each a_i is either* accept *or* deny*. Then, for each header $x \in \{0,1\}^W$ such that $x \in R$, the first TCAM entry S_j matching x is associated with $a_j =$ accept; and likewise, for each $x \notin R$, either the first TCAM entry S_j matching x is associated with $a_j =$ deny, or no TCAM entry matches x (we assume a default action of* deny*). 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 $opt_p(R) = |OPT_p(R)|$. Let $opt(R)$ be the *range expansion* of R, or just the *expansion* of R for short. Likewise let $opt_p(R)$ be 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 $r^e(W) = \max\{opt(R) \mid R = [0, y] \vee R = [y, 2^W - 1]\}$. Analogously, we define the maximum expansion with prefix TCAM entries to be $r_p(W) = \max_R opt_p(R)$, and for extremal ranges $r_p^e(W) = \max\{opt_p(R) \mid R = [0, y] \vee R = [y, 2^W - 1]\}$.

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. Extremal 1-D Ranges

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

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. The range expansion of left-extremal ranges is the same as that of right-extremal ranges. In this section, we consider only left-extremal ranges.

Likewise, 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 its subtree. For simplicity, we will therefore only consider extremal ranges.

A. Prefix Encoding Vs. 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 range expansion of R is exactly the prefix range expansion of R, i.e.*

$$opt_p(R) = opt(R). \quad (1)$$

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

As $\Phi_p \subseteq \Phi$, we trivially get $opt_p(R) \geq opt(R)$. Therefore we need to prove that $opt_p(R) \leq opt(R)$. Consider all the encoding schemes in Φ that encode the extremal range R in the minimal number of entries. Among them, consider the schemes with the minimal number of non-prefix entries, and in this subset, the schemes with the minimal number of $*$s in their non-prefix entries. Let $\phi = (S_1 \rightarrow a_1, \ldots, S_n \rightarrow a_n) \in \Phi$ be such a minimal encoding scheme. We will 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 have that $\phi \in \Phi_p$ is the required prefix encoding scheme.

Otherwise, among the non-prefix TCAM entries of ϕ, we look at the index of the left-most * in each entry. We then consider the entry with the minimal index among these indices. If there are several non-prefix entries with the same index of their left-most *, we consider the last one. We denote this entry by $S \rightarrow a$ such that $S = (s_1, \ldots, s_W) \in \{0, 1, *\}^W$ and distinguish two different cases depending on the action $a \in \mathcal{A} = \{\text{accept}, \text{deny}\}$. Let $j \in [1, W]$ be the minimal index such that $s_j = *$. Further, let $k \in [1, n]$ be the index of this TCAM entry such that $S_k = S$ and $a_k = a$.

We first assume that $a =$ accept. The case $a =$ deny is similar and is discussed in [19]. For this range $R = [0, y]$, we compare the first $j - 1$ symbols of y and S. By definition of j, we have that $\forall i \in [1, (j-1)], s_i \in \{0, 1\}$ and therefore $y_1 \ldots y_{j-1}$, $s_1 \ldots s_{j-1}$ are both binary strings. The proof now splits into several cases:

(i) We have $s_1 \ldots s_{j-1} > y_1 \ldots y_{j-1}$. In this case, the entry accepts strings which are not in the range, and therefore have been denied earlier on. Therefore, an equivalent encoding of R, with one less entry, would be to remove the entry $S_k \rightarrow$ accept. This is a contradiction to the selection of ϕ.

(ii) We have $s_1 \ldots s_{j-1} < y_1 \ldots y_{j-1}$. In this case, one can replace $S_k \rightarrow$ accept with $s_1 \ldots s_{j-1}(*)^{W-j+1} \rightarrow$ accept, to get an encoding of R with less non-prefix entries.

(iii) We have $s_1 \ldots s_{j-1} = y_1 \ldots y_{j-1}$ and $y_j = 0$. In this case, replace $S_k \rightarrow$ accept with $s_1 \ldots s_{j-1} 0 s_{j+1} \ldots s_W \rightarrow$ accept.

(iv) We have $s_1 \ldots s_{j-1} = y_1 \ldots y_{j-1}$ and $y_j = 1$, and there exists an entry S_ℓ that begins with $s_1 \ldots s_{j-1}0$. If the entry S_ℓ is of the form $S_\ell \rightarrow$ deny, deleting the entry ℓ would leave us with a more efficient encoding of R. If the entry S_ℓ is of the form $S_\ell \rightarrow$ accept, change the encoding of R by removing the entry S_ℓ, changing S_k to $s_1 \ldots s_{j-1} 1 s_{j+1} \ldots s_W \rightarrow a_k$, and add as a first entry the entry $s_1 \ldots s_{j-1} 0 (*)^{W-j} \rightarrow$ accept.

(v) Finally, we have $s_1 \ldots s_{j-1} = y_1 \ldots y_{j-1}$ and $y_j = 1$, and no entry that begins with $s_1 \ldots s_{j-1}0$ exists. Let A denote the set of 2^{W-j} strings that begin with $s_1 \ldots s_{j-1}0$. If S_k is not the first matching entry for any string in A then we can change S_k to be $s_1 \ldots s_{j-1} 1 s_{j+1} \ldots s_W \rightarrow$ accept. We denote by ℓ the index of the first entry in $S_{k+1} \rightarrow a_{k+1}, \ldots, S_n \rightarrow a_n$ that at least one of the strings in A matches. As described below, sometimes such an entry does not exist. There are two cases:

(v.a) The entry S_ℓ is of the form $S_\ell \rightarrow$ accept. In this case, since there is no entry (anywhere) that begins with $s_1 \ldots s_{j-1}0$, it must be that the index of the leftmost $*$ in S_ℓ is at most j. As the entry k is the non-prefix entry with the minimal leftmost $*$, and the last non-prefix entry among all the non-prefix entries with $*$ in place j, we have that S_ℓ is a prefix entry, of the form $s_1 \ldots s_r (*)^{W-r}$ with $r < j$. Then, for every string in A, and in particular for such strings that are first matched by S_k, the first matching entry in $S_{k+1} \rightarrow a_{k+1}, \ldots, S_n \rightarrow a_n$ is S_ℓ. Therefore, changing S_k to be $s_1 \ldots s_{j-1} 1 s_{j+1} \ldots s_W \rightarrow a_k$ produces an encoding of ϕ with less *s in the non-prefix entries - strings that begin with $s_1 \ldots s_{j-1}0$ will be accepted by S_ℓ.

(v.b) The entry S_ℓ is of the form $S_\ell \rightarrow$ deny or it does not exist, i.e. any string in A is not matched by any of the entries after S_k. In [19], we show that in both cases we can change S_k to be $s_1 \ldots s_{j-1} (*)^{W-j+1} \rightarrow$ accept while still encoding R. This decreases the number of non-prefix entries in the encoding. ∎

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 tree describing the entire space $[0, 2^W - 1]$. Each such subtree T corresponds to all binary strings starting with a particular prefix $x(T)$. That is, 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 T*.

For a subtree T we define $A(T)$ to be an optimal prefix encoding of T in which the last entry is of the form $c(T) \rightarrow$ accept, and let $n_A(T)$ be the number of entries in this encoding. If there are more than one possible optimal encodings with such last entry, $A(T)$ is an arbitrary one of them. Similarly, let $D(T)$ be an optimal prefix encoding of T with last entry $c(T) \rightarrow$ deny, and let $n_D(T)$ be the number of entries in such an optimal encoding.

Example 1. *If a subtree T satisfies $T \subseteq R$ then T can be encoded by $A(T) = (c(T) \rightarrow accept)$ in $n_A(T) = 1$ entries or by $D(T) = (c(T) \rightarrow accept, c(T) \rightarrow deny)$ with $n_D(T) = 2$. Likewise, if $T \subseteq R^c$ then T can be encoded by $A(T) = (c(T) \rightarrow deny, c(T) \rightarrow accept)$ with $n_A(T) = 2$ or by $D(T) = (c(T) \rightarrow deny)$ with $n_D(T) = 1$.*

If a subtree T contains a single input header (i.e. $|T| = 1$), then either $T \subseteq R$ or $T \subseteq R^c$. Thus $A(T)$, $n_A(T)$, $D(T)$, and $n_D(T)$ can be computed as in Example 1. In preparation for our dynamic programming algorithm, the following propositions shows how we can compute $A(T)$, $n_A(T)$, $D(T)$, and $n_D(T)$ for $|T| \geq 2$ based on the corresponding value for the left and the right subtrees of T. They also relate the value of $opt_p(R)$ and $n_D(T)$ for the complete tree T. Their simple proofs can be found in [19].

Proposition 1. *Let T be the complete tree of $[0, 2^W - 1]$ (i.e. $c(T) = (*)^W$). The prefix range expansion of a range R is $n_D(T) - 1$, i.e.*

$$opt_p(R) = n_D(T) - 1. \tag{2}$$

Proposition 2. *Let T a subtree such that $|T| \geq 2$. Let L_T, R_T be the left and right subtrees of T, respectively. Then,*

$$n_A(T) = \min\{n_A(L_T) + n_A(R_T) - 1, n_D(L_T) + n_D(R_T)\}$$
$$n_D(T) = \min\{n_A(L_T) + n_A(R_T), n_D(L_T) + n_D(R_T) - 1\}.$$

Proposition 3. *Let T be a subtree. Then, $n_D(T) \leq n_A(T) + 1$ and $n_A(T) \leq n_D(T) + 1$, i.e. $|n_A(T) - n_D(T)| \leq 1$.*

Based on Proposition 2, we suggest a simplified version of a dynamic-programming algorithm presented in [11] to compute an optimal encoding of any extremal range. Our suggested version is original in several ways. First, we demonstrate its *optimality* among all encoding schemes rather than just among prefix schemes. Second, it is *significantly easier to compute*. This is because we only consider the subtrees $T_i = y_1 \dots y_{W-i}(*)^i$ (for $i \in [1, W]$). In each step of the algorithm we calculate the parameters of a subtree based on its left and right subtrees, while the parameters of one of these two subtrees can be obtained immediately from Example 1. The full details can be found in [19].

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

C. The 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 $R = [0, y] = \{(0)^W, \dots, y_1 \dots y_W\}$. 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 $n_A(T) - n_D(T) \in \{-1, 0, 1\}$ for a subtree T. The state $A = (a, a + 1)$ represents a subtree T with $n_A(T) + 1 = n_D(T)$, the state $B = (b, b)$ represents a subtree T with $n_A(T) = n_D(T)$, and the state $C = (c+1, c)$ represents a subtree T with $n_A(T) = n_D(T) + 1$.

The input to the DFA would be the binary string $y_1 \dots y_W$ *in a right to left order*. The starting state is A and the transition function $\delta : Q \times \Sigma \rightarrow Q$ 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 from the computation of this DFA. To do so, we suggest the following:

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

Proof Outline: We calculate the value of $n_D(T_i)$ by induction on i. We show its dependency on the number of transitions of the two specific forms in the DFA. Finally, by Proposition 1, $opt_p(R) = n_D(T) - 1 = n_D(T_W) - 1$, which leads to the result. ∎

D. Average Range Expansion For Extremal Ranges

We now use the DFA of Section III-C to derive a closed-form formula for the average range expansion of an extremal range formally defined as

$$G(W) = \mathbb{E}_{y:\, 0 \leq y \leq 2^W - 1}\left(opt([0, y])\right)$$

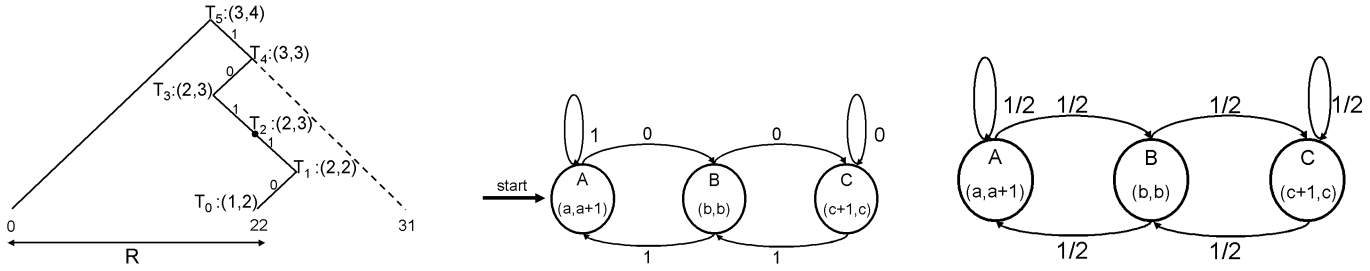
Theorem 3. *The average extremal range expansion function $G(W)$ satisfies*

$$G(W) = \frac{4}{9} + \frac{W}{3} + \frac{4}{9} \cdot \left(\frac{1}{2}\right)^W \text{ if } W \text{ is odd, and}$$
$$G(W) = \frac{4}{9} + \frac{W}{3} + \frac{5}{9} \cdot \left(\frac{1}{2}\right)^W \text{ if } W \text{ is even.} \tag{3}$$

Proof Outline: To calculate $G(W)$, we derive a Markov chain from the DFA of Section III-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. We then find that $G(W) = 1 + \frac{1}{2} \cdot \sum_{i=0}^{W-1} \left(1 - (P^i)_{(1,1)}\right)$, where P is the transition matrix. Since for any j, $(P^{2j-1})_{(1,1)} = (P^{2j})_{(1,1)} = \frac{1}{3} + \frac{2}{3} \cdot \left(\frac{1}{2}\right)^{2j}$, we end up solving simple recursive equations and obtain the needed results. ∎

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

By [17], the worst case expansion for an extremal range is $r^e(W) = r_p^e(W) = \left\lceil \frac{W+1}{2} \right\rceil$. Thus clearly $G(W) \leq \left\lceil \frac{W+1}{2} \right\rceil$. Theorem 3 and its corollary below show that the average encoding length is only about $2/3$ of the worst case.



(a) Illustration of the algorithm (b) A Deterministic Finite Automaton (DFA) (c) The corresponding Markov Chain

Fig. 2. Illustration of the algorithm results for the extremal range $R = [0, 22]$ from Example 2. The parameters $(n_A(T_i), n_D(T_i))$ of each tree $T_i \in \{T_0, \ldots, T_W\}$ are illustrated. The parameter $n_A(T_i)$ is the number of entries in the smallest encoding of T_i with a last entry of the form $c(T_i) \rightarrow$ accept. Likewise, $n_D(T_i)$ is the size of the smallest encoding with a last entry $c(T_i) \rightarrow$ deny. The smallest encoding of R has $opt(R) = n_D(T_5) - 1 = 4 - 1 = 3$ entries. (b) presents a Deterministic Finite Automaton (DFA), as discussed in Section III-C. It has three states representing the three possible values of $(n_A(T) - n_D(T)) \in \{-1, 0, 1\}$ in a subtree T. (c) shows the corresponding Markov Chain of the DFA with the same 3 states.

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

$$\lim_{W \to \infty} \frac{G(W)}{W} = \frac{1}{3}. \tag{4}$$

IV. Bounds on Worst-Case Expansion

A. General 1-D Ranges

The next theorem shows that the upper-bound on the maximum range expansion $r(W) \leq W$ is actually tight. Unlike the limited result from [17] that shows its *tightness among only prefix encoding schemes*, we show its *tightness among all TCAM encoding schemes*. Incidentally, *this theorem answers the open question left in [17].*

Theorem 5. *For all* $W \geq 1$, *the maximum range expansion satisfies*

$$r(W) = r_p(W) = W. \tag{5}$$

Proof Outline: See [19] for the full proof. We first suggest *a new analytical tool* called a *conflicting set of pairs*. Intuitively, a conflicting set of pairs of size n is composed of n pairs of points, each with one point within the range and one outside the range. We show that the pairs are *pairwise conflicting*, so that we cannot encode together two points within the range or alternatively two points outside the range from two different pairs. We show that a range R (either $1D$ or $2D$) with a conflicting set of pairs of size n cannot be encoded in less than n TCAM entries. It is known from [17] that $r(W) \leq r_p(W) \leq W$. To show that $r(W) \geq W$, we exhibit a range R with a conflicting set of pairs of size W. ∎

B. General 2-D Ranges

We now consider the encoding of two-dimensional ranges. The input here is a pair of strings (a, b). A two-dimensional range is a product of two one-dimensional ranges $R_1 \times R_2$, and the encoding of such a range should accept exactly the pairs of strings (a, b) such that $a \in R_1$ and $b \in R_2$.

We generalize the definition of $r(W)$ to multi-dimensional ranges, and define $r^d(W)$ as the maximum expansion of a 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 $r_p^d(W)$ (respectively $r_p^{e,d}(W)$) be the maximum expansion of a (an extremal) d-dimensional range when we use only prefix encodings.

Theorem 6. *The worst-case expansion of a two-dimensional classification rule satisfies,*

$$r^2(W) = r_p^2(W) = 2W. \tag{6}$$

Proof Outline: We first show that $r^2(W) \leq r_p^2(W) \leq 2W$. To do so, we consider for a two-dimensional range $R^2 = R_x \times R_y$, two possible encodings. First, we can negatively encode the complementary of R_x (with don't-cares for the second field) and then positively R_y. Alternatively, we can start by encoding negatively the complementary of R_y and then positively R_x. We show that at least one of these encodings has the required expansion. Finally, to show the tightness of this lower bound, we suggest a two-dimensional range R with a conflicting set of pairs of size $2W$. ∎

C. Extremal 2-D Ranges

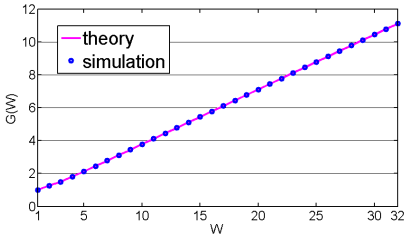
As in the case of one-dimensional ranges, the upper bound for general two-dimensional ranges from Theorem 6 can be improved when only extremal ranges are considered.

Theorem 7. *The worst-case expansion of a two-dimensional extremal classification rule satisfies*

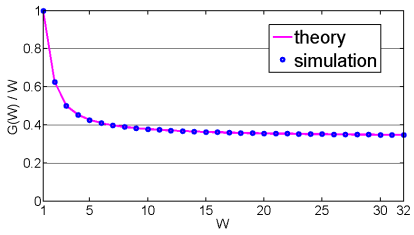
$$2 \cdot \left\lceil \frac{W+1}{2} \right\rceil - 1 \leq r^{e,2}(W) \leq r_p^{e,2}(W) \leq W + 1. \tag{7}$$

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

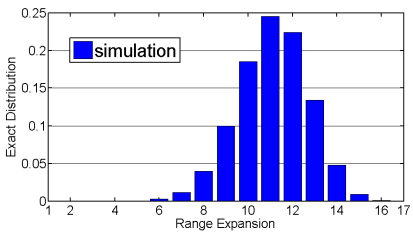
Proof Outline: See [19] for the full proof. We again consider the same two possible encodings suggested earlier for non-necessarily extremal two-dimensional ranges. We prove that for two-dimensional extremal ranges the best one achieves the improved upper bound. Later, we exhibit a range with a conflicting set of pairs of the required size to get the lower bound. ∎



(a) The average extremal range expansion $G(W)$ presented in Theorem 3.



(b) The normalized average extremal range expansion $G(W)/W$. We can see that indeed $\lim_{W\to\infty}\frac{G(W)}{W}=\frac{1}{3}$ as stated by Corollary 4.



(c) Extremal range expansion distribution for $W=32$. The minimal expansion is 1 and the maximal expansion is $\left\lceil\frac{W+1}{2}\right\rceil=17$.

Fig. 3. Simulations of extremal range expansion

V. Experimental Results

A. One-Dimensional Extremal Ranges

We conduct simulations to examine the results of the average range expansion for extremal ranges presented in Section III-D. Fig. 3(a) presents the function $G(W)$ for $W\in[1,32]$. For each value of W, the average expansion is calculated based on 2^W extremal ranges. 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 2 entries.

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

Last, Fig. 3(c) presents the distribution of the extremal range expansion for $W=32$. The minimal expansion is of course 1 and the maximal expansion is $\left\lceil\frac{W+1}{2}\right\rceil=17$, both with negligible probability. The most popular expansion is 11, and there are about a billion $(1,053,445,120)$ different extremal ranges with such an expansion, out of about 4 billion (2^{32}) left-extremal ranges.

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 suggested encoding scheme for two-dimensional ranges from Section IV (with an improved worst-case expansion of $2W$) in comparison with other well-known encoding schemes such as the Binary Prefix encoding [16], the SRGE encoding [9] and the external encoding for two-dimensional ranges from [17].

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 with 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 previously used in [1], [9], [10]. In this database, the source port and the destination port are W-bit fields (with $W=16$). The rules might include ranges in these fields. 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* (51,055 rules out of 57,146).

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

D. Effectiveness on Real-life Packet Classifiers

Fig. 4(a) presents the total expansion of the two-dimensional ranges in twelve artificial classifiers generated by the ClassBench benchmark tool [25] and on the union of the 120 real-life rule files. It compares the expansion of the suggested encoding scheme for two-dimensional ranges (with the upper bound of $2W$) versus Binary Prefix encoding [16] and SRGE encoding [9]. For the classifier *fw4*, for instance, the total expansion is 33,774 entries in comparison with 154,813 and 153,691 entries. An improvement of 78.2% and 78.0%, respectively. Likewise, for the real-life files, the improvement is 73.4% in comparison with Binary Prefix.

Fig. 4(b) compares the total expansion of all rules in these classifiers in the regular TCAM architecture using Binary Prefix and SRGE (illustrated in the two left bars in each group of three) and in the suggested joint TCAM architecture from Fig. 1 (in the right bar). In this simulation, we choose to encode all the two-dimensional ranges in the second part of the architecture using also *deny* 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 *accept* entries), is exactly

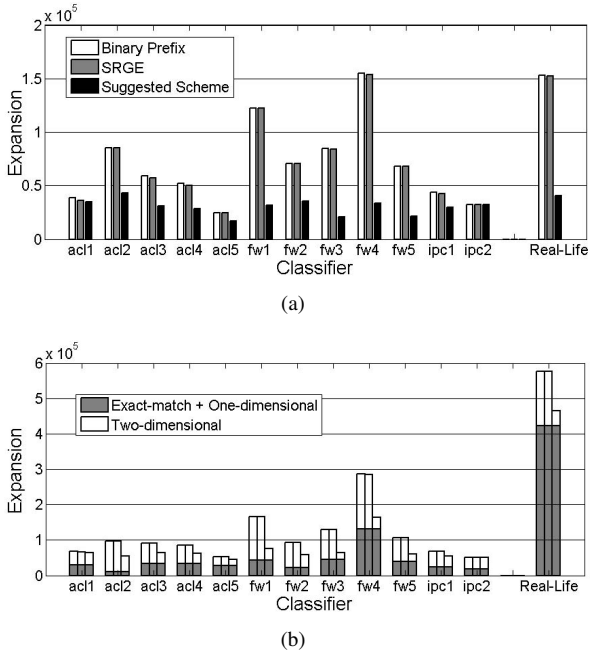


Fig. 4. Effectiveness of the suggested encoding scheme and the suggested joint TCAM architecture (illustrated in Fig. 1) on twelve artificial classifiers generated by ClassBench benchmark tool and on a real-life database. For each classifier, the two left bars present the expansion of Binary Prefix and of SRGE, while the third bar illustrates our suggested solution. In (a), we compare the total expansion of the two-dimensional ranges in the classifiers. In (b), we examine the expansion using the joint TCAM architecture when the two-dimensional ranges are encoded in the second (modified) TCAM, i.e. the white bars correspond to (a).

as in Binary Prefix encoding. Thus, 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% in comparison with Binary Prefix. This essentially serves as a proof of concept to our joint TCAM architecture.

VI. Conclusion

In this paper, we took a first step towards finding an optimal TCAM encoding algorithm for all classification rules. We presented an encoding algorithm that is optimal for all possible *generalized extremal rules*, which represent 89% of all non trivial rules in a typical real-life classification database. We also obtained new tight bounds on the worst case for general classification rules, both for one-dimensional and two-dimensional ranges. Finally we presented a novel combined TCAM architecture, composed of a regular TCAM and a modified TCAM, which can provide a guaranteed improved expansion for the tough classification rules.

VII. Acknowledgment

We would like to thank Noam Nisan, Danny Raz, Alex Shpiner and Aran Bergman for their helpful participation and suggestions. We would also like to acknowledge Anat Bremler-Barr for kindly accepting to run several simulations.

This work was partly supported by the European Research Council Starting Grant No. 210389, by the Israel Science Foundation grants No. 822/10 and 1241/12, by the United States-Israel Binational Science Foundation project No. 2006204, by the German-Israeli Foundation for Scientific Research and Development, by the Israeli Centers of Research Excellence (I-CORE) program No. 4/11, by the Intel ICRI-CI Center, by the Hasso Plattner Center for Scalable Computing and by the Israel Ministry of Science and Technology. Ori Rottenstreich is the Google Europe Fellow in Computer Networking, an Andrew and Erna Finci Viterbi Fellow and a Jacobs-Qualcomm Fellow.

References

[1] D. E. Taylor, "Survey and taxonomy of packet classification techniques," *ACM Comput. Surv.*, vol. 37, no. 3, pp. 238–275, 2005.
[2] G. Varghese, *Network Algorithmics*. Morgan Kaufmann, 2005.
[3] J. Chao and B. Liu, *High Performance Switches and Routers*. Wiley, 2007.
[4] J. Naous, D. Erickson, A. Covington, G. Appenzeller, and N. McKeown, "Implementing an OpenFlow switch on the NetFPGA platform," in *ACM ANCS*, 2008.
[5] NetLogic Microsystems. [Online]. Available: www.netlogicmicro.com/
[6] Renesas. [Online]. Available: www.renesas.com/
[7] P. Gupta and N. McKeown, "Packet classification on multiple fields," in *ACM SIGCOMM*, 1999.
[8] S. Singh, F. Baboescu, G. Varghese, and J. Wang, "Packet classification using multidimensional cutting," in *ACM SIGCOMM*, 2003.
[9] A. Bremler-Barr and D. Hendler, "Space-efficient TCAM-based classification using gray coding," *IEEE Trans. Computers*, vol. 61, no. 1, 2012.
[10] K. Lakshminarayanan, A. Rangarajan, and S. Venkatachary, "Algorithms for advanced packet classification with ternary CAMs," in *ACM SIGCOMM*, 2005.
[11] S. Suri, T. Sandholm, and P. R. Warkhede, "Compressing two-dimensional routing tables," *Algorithmica*, vol. 35, no. 4, pp. 287–300, 2003.
[12] T. Sasao, "On the complexity of classification functions," in *ISMVL*, 2008.
[13] Y.-K. Chang, C.-I. Lee, and C.-C. Su, "Multi-field range encoding for packet classification in TCAM," in *IEEE Infocom Mini-Conference*, 2011.
[14] C. R. Meiners, A. X. Liu, and E. Torng, "Bit weaving: A non-prefix approach to compressing packet classifiers in TCAMs," *IEEE/ACM Trans. Networking*, vol. 20, no. 2, pp. 488–500, 2012.
[15] E. Spitznagel, D. E. Taylor, and J. S. Turner, "Packet classification using extended TCAMs," in *ICNP*, 2003.
[16] V. Srinivasan, G. Varghese, S. Suri, and M. Waldvogel, "Fast and scalable layer four switching," in *ACM SIGCOMM*, 1998.
[17] O. Rottenstreich, R. Cohen, D. Raz, and I. Keslassy, "Exact worst-case TCAM rule expansion," *IEEE Trans. Computers*, 2012.
[18] W. Jiang and V. K. Prasanna, "Scalable packet classification on FPGA," *IEEE Trans. VLSI Syst.*, vol. 20, no. 9, pp. 1668–1680, 2012.
[19] O. Rottenstreich, I. Keslassy, A. Hassidim, H. Kaplan, and E. Porat, "On finding an optimal TCAM encoding scheme," Technion, Tech. Rep. TR12-04, 2012. [Online]. Available: http://webee.technion.ac.il/~isaac/papers.html
[20] B. Schieber, D. Geist, and A. Zaks, "Computing the minimum DNF representation of Boolean functions defined by intervals," *Discrete Applied Mathematics*, vol. 149, no. 1-3, pp. 154–173, 2005.
[21] R. Wei, Y. Xu, and H. J. Chao, "Block permutations in boolean space to minimize TCAM for packet classification," in *IEEE Infocom Mini-Conference*, 2012.
[22] Y. Ma and S. Banerjee, "A smart pre-classifier to reduce power consumption of TCAMs for multi-dimensional packet classification," in *ACM SIGCOMM*, 2012.
[23] C. R. Meiners, A. X. Liu, and E. Torng, "Topological transformation approaches to TCAM-based packet classification," *IEEE/ACM Trans. Networking*, vol. 19, no. 1, pp. 237–250, 2011.
[24] H. Che, Z. Wang, K. Zheng, and B. Liu, "Dres: Dynamic range encoding scheme for TCAM coprocessors," *IEEE Trans. Computers*, vol. 57, no. 7, pp. 902–915, 2008.
[25] D. E. Taylor and J. S. Turner, "ClassBench: a packet classification benchmark," in *IEEE Infocom*, 2005.