On Finding an Optimal TCAM Encoding Scheme for Packet Classification

Ori Rottenstreich*, Isaac Keslassy*, Avinatan Hassidim‡, Haim Kaplan§, Ely Porat¶
*Technion, {or@tx,isaac@ee}.technion.ac.il
‡Google Israel and Bar-Ilan University, avinatanh@google.com
§Tel Aviv University, haimk@post.tau.ac.il
¶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. We focus here on the common scenario where the possible actions are either to *accept* or *deny* the packet.

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). 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]–[17].

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.

Our first contribution is a theoretical one. 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 sub-tree of this tree. For instance, within a tree with 64 leaves, $[5, 7]$ is extremal for the sub-tree 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

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	-	[18]	$\surd$	-	$\surd$	-
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	-	[18]	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], [19]–[21]	$\lceil \frac{W+1}{2} \rceil$	$\lceil \frac{W+1}{2} \rceil$	$\boxed{\mathbf{W+1}}$	$\boxed{\mathbf{W+1}}$	W	W	$\boxed{\mathbf{2W}}$	$\boxed{\mathbf{2W}}$
-	-	-	[21]	$\lceil \frac{W+1}{2} \rceil$	$\lceil \frac{W+1}{2} \rceil$	$\boxed{\mathbf{W+1}}$	$\boxed{\mathbf{W+1}}$	W	$\boxed{\mathbf{W}}$	$\boxed{\mathbf{2W}}$	$\boxed{\mathbf{2W}}$

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 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 often added encoding constraints, e.g. by constraining the encoding to be prefix, and therefore achieved sub-optimal encoding.

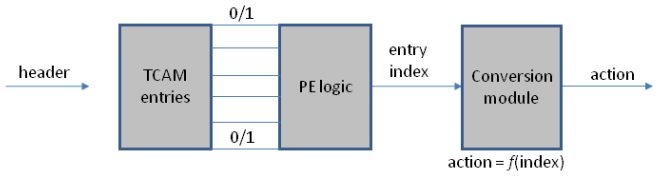
This contribution is especially interesting in that the set of generalized extremal ranges is a significant set in practice. To estimate the potential impact of our results we consider a union of 120 real-life classification databases from [10]. It contains 214,941 rules. We find that 97.2% *of these 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.

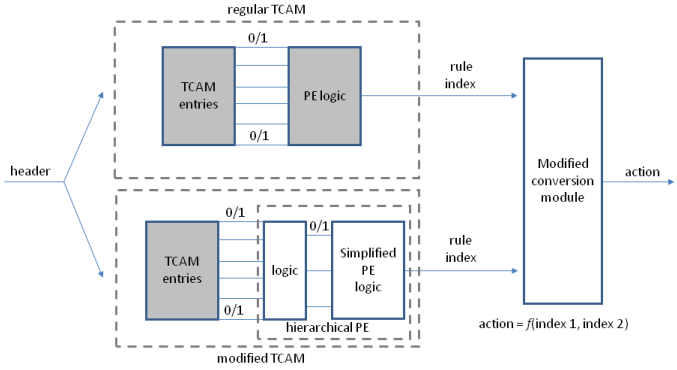
Our second main contribution is that *we find the exact worst-case expansions of both one-dimensional and two-dimensional ranges* (last rows of Table I(B)). In particular, we first present a simple algorithm that is optimal for general one-dimensional ranges in the worst case. Specifically, we first show that it encodes any range in $[0, 2^W - 1]$ in at most W rules. This algorithm uses either a set of *accept* rules that accept the range, or a set of *deny* rules that deny the complement of the range together with a last default *accept* rule that accepts everything else. We also 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 $\lceil (W + 1)/2 \rceil$ [21].

In addition, we also present an algorithm that is optimal in the worst case for general two-dimensional ranges. In other words, we analyze the two-dimensional scenario in which a classification rule includes two ranges, one for the source port and the other for the destination port. In this case we present an algorithm that encodes any two-dimensional range with at most $2W$ entries, where W is the length of each of the two fields. We also exhibit 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



(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.

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. Also, while the stand-alone modified TCAM architecture is known in the literature [22], it may waste practical resources when applied on the set of all rules. Instead, our new architecture uses a regular TCAM for the simple rules, and only spends the additional logic on the complex ones.

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], [23]. 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 [22], 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%.

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* [18]. 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: $(0001 \rightarrow \text{accept}, 001* \rightarrow \text{accept}, 01** \rightarrow \text{accept}, 10** \rightarrow \text{accept}, 110* \rightarrow \text{accept}, 1110 \rightarrow \text{accept}, (**** \rightarrow \text{deny}))$.

A first improvement of the $2W-2$ result used non-prefix TCAM encoding and a connection to Boolean DNF (disjunctive normal form) to show an upper bound of $2W-4$ [24]. A second improvement used Gray codes instead of binary codes to reduce the worst-case TCAM size for any range from $2W-2$ to $2W-4$ as well [9].

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 [19], [20]. 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]–[17].

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, [21] 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.*

Recently, a two-level TCAM architecture was suggested to reduce power consumption in packet classification [25].

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.

Paper Organization: We start with preliminary definitions in Section II. Then, in Section III we consider one-dimensional extremal ranges. We suggest an optimal encoding scheme for any given extremal range and calculate the average range expansion over extremal ranges. In Section IV we present the conflicting set of pairs, a new analytical tool for proving lower bounds on the range expansion. Next, in Section V we provide new tight bounds on the worst-case expansion of one-dimensional and two-dimensional ranges. Last, we examine our theoretical results and evaluate the suggested encoding schemes in Section VI.

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 $r_1 = 0$ we call the range a left-extremal range*, and when $r_2 = 2^W - 1$ we call the range a* right-extremal range*. An* extremal range *is either a left-extremal range or a right-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 *TCAM expansion* of R, or just the *expansion* of R for short. Likewise let $opt_p(R)$ be the *prefix TCAM 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.

As defined in Definition 1, 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 TCAM 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 TCAM expansion of R is exactly the prefix TCAM expansion of R, i.e.*

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

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 according to 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 we discuss it shortly at the end of the proof. 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 rule accepts strings which are not in the range, and therefore have been denied earlier on. Therefore, an equivalent encoding of R, with one less rule, would be to remove the rule $S_k \rightarrow$ '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 rows.

(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 a rule S_ℓ that begins with $s_1 \ldots s_{j-1} 0$. If the rule S_ℓ is of the form $S_\ell \rightarrow$ 'deny', deleting the rule ℓ would leave us with a more efficient encoding of R. If the rule S_ℓ is of the form $S_\ell \rightarrow$ 'accept', change the encoding of R by removing the rule 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 rule the rule $s_1 \ldots s_{j-1} 0 (*)^{W-j} \rightarrow$ 'accept'.

(v) Finally, we have $s_1 \ldots s_{j-1} = y_1 \ldots y_{j-1}$ and $y_j = 1$, and no rule that begins with $s_1 \ldots s_{j-1} 0$ exists. Consider now some string $x = s_1 \ldots s_{j-1} 0 x_{j+1} \ldots x_W$ which its first match is in S_k (if no such x exists change S_k to be $s_1 \ldots s_{j-1} 1 s_{j+1} \ldots s_W \rightarrow$ 'accept'). There are two cases:

(v.a) The smallest $\ell > k$ such that $x \in S_\ell$ is of the form $S_\ell \rightarrow$ 'accept'. In this case, since there is no rule (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 rule k is the non-prefix rule with the minimal leftmost $*$, and the last non-prefix rule among all the non-prefix rules with $*$ in place j, we have that S_ℓ is a prefix rule, of the form $s_1 \ldots s_r (*)^{W-r}$ with $r < j$. But for every string y that begins with $s_1 \ldots s_{j-1} 0$ the first rule in $S_{k+1} \rightarrow a_{k+1}, \ldots, S_n \rightarrow a_n$ which affects y 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 - all the strings that begin with $s_1 \ldots s_{j-1} 0$ will be accepted by S_ℓ.

(v.b) The smallest $\ell > k$ such that $x \in S_\ell$ is of the form $S_\ell \rightarrow$ 'deny'. Similarly to the previous case, for every string y that begins with $s_1 \ldots s_{j-1} 0$ we have that the first rule in $S_{k+1} \rightarrow a_{k+1}, \ldots, S_n \rightarrow a_n$ which affects y is S_ℓ. But since ϕ is encoded correctly, this means that $S_1 \rightarrow a_1, \ldots, S_k \rightarrow a_k$ accept every string that begins with $s_1 \ldots s_{j-1} 0$. Since there is no rule (anywhere) that begins with $s_1 \ldots s_{j-1} 0$, it must be the case that every string z that begins with $s_1 \ldots s_{j-1} 1$ has $z \in S_r$ for some $r \leq k$. Therefore, if it should be rejected by R, when reaching the k^{th} rule it was already denied. This means that 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 rules in the encoding.

The case $a =$ 'deny' is similar to the accept case. Note that the asymmetry between accept and reject which is created by the default is not an issue here, as case *(v.a)* (respectively *(v.b)*) treats the possibility that the next rule to be applied to these values is an accept (respectively a reject). In particular, the default accept rule is covered by *(v.a)*. ■

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 trie describing the entire space $[0, 2^W-1]$. Each such subtree T corresponds to all 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 binary trie T.

Proposition 1. *Let T be the complete binary trie 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.*

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

Proof: Given a range R, $D(T)$ is an optimal prefix encoding of T in $n_D(T)$ entries with a last entry of $c(T) = (*)^W \rightarrow$ 'deny'. If we omit this last entry any input header that was first matched by this entry is not matched now and the default action of 'deny' applies to it. Thus even if we omit the last entry of $D(T)$ we have a correct encoding of R so $opt_p(R) \leq n_D(T) - 1$. Likewise, if R can be encoded in $opt_p(R)$ entries, we can simply add to such encoding the entry $(*)^W \rightarrow$ 'deny' and the encoding remains correct. It follows that $opt_p(R) + 1 \geq n_D(T)$. ∎

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,*

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

Proof: We start with the first equality and show that $n_A(T) \leq \min\{n_A(L_T) + n_A(R_T) - 1, n_D(L_T) + n_D(R_T)\}$.

We can get an encoding of T as follows. We concatenate the encodings $A(L_T)$ and $A(R_T)$. Then we remove the last entry $c(L_T) \rightarrow$ 'accept' of the first encoding, and the last entry $c(R_T) \rightarrow$ 'accept' of the second encoding, and add the entry $c(T) \rightarrow$ 'accept' as the last in the new encoding. It is easy to verify that we get a correct encoding of T with $n_A(L_T) + n_A(R_T) - 1$ entries, and therefore $n_A(T) \leq n_A(L_T) + n_A(R_T) - 1$

Alternatively, we can concatenate the encodings $D(L_T)$ and $D(R_T)$ without their last entries and add two new entries at the end $c(T) \rightarrow$ 'deny' followed by $c(T) \rightarrow$ 'accept'. Again, it is easy to verify that we get a correct encoding of T with $n_D(L_T) + n_D(R_T)$ entries so $n_A(T) \leq n_D(L_T) + n_D(R_T)$. This completes the proof that $n_A(T) \leq \min\{n_A(L_T) + n_A(R_T) - 1, n_D(L_T) + n_D(R_T)\}$.

We now want to show that $n_A(T) \geq \min\{n_A(L_T) + n_A(R_T) - 1, n_D(L_T) + n_D(R_T)\}$. To show that, we prove that any optimal prefix TCAM encoding $A(T)$ of T with a last entry $c(T) \rightarrow$ 'accept' satisfies that $n_A(T) \geq n_A(L_T) + n_A(R_T) - 1$ or $n_A(T) \geq n_D(L_T) + n_D(R_T)$.

If $n_A(T) = 1$, we clearly have that $T \subseteq R$ and thus also $n_A(L_T) = n_A(R_T) = 1$ and the first of the last two inequalities holds. So we may assume that $n_A(T) \geq 2$.

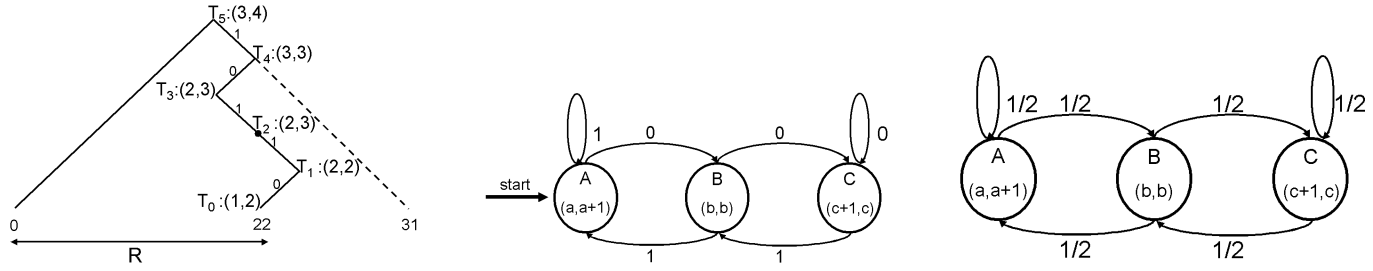
All the entries in $A(T)$ start with the string $x(T)$. We divide the entries of $A(T)$ into three disjoint subsets according to their value in the $|x(T)| + 1$ leftmost symbol and define, $A(T)_0 = \{S = s_1 \dots s_W \rightarrow a \mid S \in A(T), s_{|x(T)|+1} = 0\}$, $A(T)_1 = \{S = s_1 \dots s_W \rightarrow a \mid S \in A(T), s_{|x(T)|+1} = 1\}$ and $A(T)_* = \{S = s_1 \dots s_W \rightarrow a \mid S \in A(T), s_{|x(T)|+1} = *\}$. Since $A(T)$ is a prefix encoding, any entry in $A(T)_*$ is of the form $c(T) \rightarrow a$ for $a \in \{$'accept', 'deny'$\}$. Thus the entries in $A(T)_*$ must be the last entries of $A(T)$ because any entry which follows them is redundant.

We distinguish two cases according to the size of $A(T)_*$. If $|A(T)_*| = 1$ then $A(T)_*$ is composed of the last entry of $A(T)$, $c(T) \rightarrow$ 'accept' and $|A(T)_0| + |A(T)_1| = n_A(T) - 1$. We encode L_T by adding a last entry $c(L_T) \rightarrow$ 'accept' to the entries in $A(T)_0$ and encode R_T by adding a last entry $c(R_T) \rightarrow$ 'accept' to the entries in $A(T)_1$. By the correctness of $A(T)$, these are correct encodings of L_T and R_T, both with a last entry with the action 'accept'. We then have that $n_A(L_T) + n_A(R_T) \leq |A(T)_0| + |A(T)_1| + 2 = n_A(T) - 1 + 2 = n_A(T) + 1$ and $n_A(T) \geq n_A(L_T) + n_A(R_T) - 1$.

If $|A(T)_*| = 2$, then the additional entry in $A(T)_*$ is $c(T) \rightarrow$ 'deny' which is the $(n_A(T) - 1)^{\text{th}}$ entry of $A(T)$. Then, we encode L_T by adding a last entry $c(L_T) \rightarrow$ 'deny' to the entries in $A(T)_0$ and encode R_T by adding a last entry $c(R_T) \rightarrow$ 'deny' to the entries in $A(T)_1$ and we get that $n_D(L_T) + n_D(R_T) \leq |A(T)_0| + |A(T)_1| + 2 = n_A(T) - 2 + 2 = n_A(T)$ and $n_A(T) \geq n_D(L_T) + n_D(R_T)$.

We got that either $n_A(T) \geq n_A(L_T) + n_A(R_T) - 1$ or $n_A(T) \geq n_D(L_T) + n_D(R_T)$ and thereby $n_A(T) \geq \min\{n_A(L_T) + n_A(R_T) - 1, n_D(L_T) + n_D(R_T)\}$. This completes the proof of the equality $n_A(T) = \min\{n_A(L_T) + n_A(R_T) - 1, n_D(L_T) + n_D(R_T)\}$.

The proof of the equality $n_D(T) = \min\{n_A(L_T) + n_A(R_T), n_D(L_T) + n_D(R_T) - 1\}$ is analogous. ∎



(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.

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

Proof: If we add the entry $c(T) \rightarrow$ 'deny' to $A(T)$ we still have a correct encoding of T and therefore $n_D(T) \leq n_A(T) + 1$. Similarly, by adding the entry $c(T) \rightarrow$ 'accept' to $D(T)$ the encoding remains a correct encoding of T so $n_A(T) \leq n_D(T) + 1$. ∎

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 $T_i = y_1 \ldots y_{W-i}(*)^i$ (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 them can be obtained immediately from Example 1.

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 $A(T), D(T), n_A(T), n_D(T)$ 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 trie of $[0, 2^W - 1]$, T_0 is a single leaf and T_W is the entire trie. An optimal encoding of R is given by the $n_D(T_W) - 1$ first entries in $D(T_W)$.*

Since $T_0 \subseteq R = [0, y] = \{(0)^W, \ldots, y_1 \ldots y_W\}$, we have that $A(T_0) = (c(T_0) \rightarrow$ 'accept'$), n_A(T_0) = 1$ and $D(T_0) = (c(T_0) \rightarrow$ 'accept'$, c(T_0) \rightarrow$ 'deny'$), n_D(T_0) = 2$, as described in Example 1. Next, we consider the subtree $T_i = y_1 \ldots y_{W-i}(*)^i$ (for $i \in [1, W]$) that contains T_{i-1} as a left or right subtree, and distinguish two cases according to the value of y_{W-i+1}. If $y_{W-i+1} = 0$, then T_i's values are calculated based on its two subtrees $L_{T_i} = T_{i-1} = y_1 \ldots y_{W-i}0(*)^{i-1}$ and $R_{T_i} = y_1 \ldots y_{W-i}1(*)^{i-1} \subseteq R^c$. Similarly, if $y_{W-i+1} = 1$, then and $L_{T_i} \subseteq R$ and $R_{T_i} = T_{i-1}$. It follows that in either case, for each considered subtree, one of its subtrees was considered in the previous step of the algorithm while the second subtree is included either in the range R or in the range complementary R^c. Thus its parameters can be computed as in Example 1.

Next, we try to understand the behavior of the values of $n_A(T_i), n_D(T_i)$ and the optimal encodings $A(T_i), D(T_i)$ for a subtree $T_i = y_1 \ldots y_{W-i}(*)^i$.

If $y_{W-i+1} = 0$ then $L_{T_i} = T_{i-1}$ and $R_{T_i} \subseteq R^c$. We then have that $A(R_{T_i}) = (c(R_{T_i}) \rightarrow$ 'deny'$, c(R_{T_i}) \rightarrow$ 'accept'$), n_A(R_{T_i}) = 2$ and $D(R_{T_i}) = (c(R_{T_i}) \rightarrow$ 'deny'$), n_D(R_{T_i}) = 1$. We can obtain $D(T_i)$ from $D(T_{i-1})$ by replacing its last entry $c(T_{i-1}) \rightarrow$ 'deny' with $c(T_i) \rightarrow$ 'deny' so $n_D(T_i) = n_D(T_{i-1})$.

To compute $A(T_i)$ and $n_A(T_i)$, we split into two subcases according to the values of $n_A(L_{T_i})$ and $n_D(L_{T_i})$. By Proposition 3, these are the only subcases possible.

Case 1: *If $n_A(L_{T_i}) + 1 = n_D(L_{T_i})$ or $n_A(L_{T_i}) = n_D(L_{T_i})$ then $n_A(T_i) = \min\{n_A(L_{T_i}) + n_A(R_{T_i}) - 1, n_D(L_{T_i}) + n_D(R_{T_i})\} = \min\{n_A(L_{T_i}) + 1, n_D(L_{T_i}) + 1\} = n_A(L_{T_i}) + 1$. Here, we can get $A(T_i)$ by replacing the last entry $c(L_{T_i}) \rightarrow$ 'accept' of $A(L_{T_i})$ by the two entries $(c(R_{T_i}) \rightarrow$ 'deny'$, c(T_i) \rightarrow$ 'accept'$)$.*

Case 2: *If $n_A(L_{T_i}) = n_D(L_{T_i}) + 1$ then $n_A(T_i) = \min\{n_A(L_{T_i}) + n_A(R_{T_i}) - 1, n_D(L_{T_i}) + n_D(R_{T_i})\} = \min\{n_A(L_{T_i}) + 1, n_D(L_{T_i}) + 1\} = n_D(L_{T_i}) + 1 = n_A(L_{T_i})$. To get $A(T_i)$, we replace the last entry $c(L_{T_i}) \rightarrow$ 'deny' of $D(L_{T_i})$ by the two entries $(c(T_i) \rightarrow$ 'deny'$, c(T_i) \rightarrow$ 'accept'$)$.*

If $y_{W-i+1} = 1$ then $L_{T_i} \subseteq R$ and $R_{T_i} = T_{i-1}$ and the analysis is symmetric to the previous case.

Here, $A(L_{T_i}) = (c(L_{T_i}) \rightarrow$ 'accept'$), n_A(L_{T_i}) = 1$, $D(L_{T_i}) = (c(L_{T_i}) \rightarrow$ 'accept'$, c(L_{T_i}) \rightarrow$ 'deny'$)$, and $n_D(L_{T_i}) = 2$. We can obtain $A(T_i)$ from $A(T_{i-1})$ by replacing its last entry $c(T_{i-1}) \rightarrow$ 'accept' by $c(T_i) \rightarrow$ 'accept' so $n_A(T_i) = n_A(T_{i-1})$.

Next, we compute $D(T_i)$ and $n_D(T_i)$ based on the values $n_A(R_{T_i})$, and $n_D(R_{T_i})$. If $n_A(R_{T_i}) = n_D(R_{T_i}) + 1$ or $n_A(R_{T_i}) = n_D(R_{T_i})$ then $n_D(T_i) = \min\{n_A(L_{T_i}) + n_A(R_{T_i}), n_D(L_{T_i}) + n_D(R_{T_i}) - 1\} = \min\{n_A(R_{T_i}) + 1, n_D(R_{T_i})+1\} = n_D(R_{T_i})+1$. Now, to get $D(T_i)$, we replace the last entry $c(R_{T_i}) \rightarrow$ 'deny' of $D(R_{T_i})$ by the two entries $(c(L_{T_i}) \rightarrow$ 'accept'$, c(T_i) \rightarrow$ 'deny'$)$.

If $n_A(R_{T_i}) + 1 = n_D(R_{T_i})$ then $n_D(T_i) = \min\{n_A(L_{T_i}) + n_A(R_{T_i}), n_D(L_{T_i}) + n_D(R_{T_i}) - 1\} = \min\{n_A(R_{T_i}) + 1, n_D(R_{T_i}) + 1\} = n_A(R_{T_i}) + 1 = n_D(R_{T_i})$. To get $D(T_i)$,

we replace the last entry $R_{T_i} \rightarrow$ 'accept' of $A(R_{T_i})$ by the two entries $(c(T_i) \rightarrow$ 'accept', $c(T_i) \rightarrow$ 'deny').

Example 2. *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* $n_A(T_0) = 1$ *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* $c(T_i) = y_1 \ldots y_{W-i}(*)^i$ *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 $R = [0, y] = \{(0)^W, \ldots, y_1 \ldots y_W\}$. 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 $n_A(T) - n_D(T) \in \{-1, 0, 1\}$ 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 $n_A(T)+1 = n_D(T)$, 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 $y_1 \ldots y_W$ *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 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 $y_W, \ldots, y_{W-i+1}$. We then use the following proposition that connects between T_i and q_i.

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* $n_A(T_i)$ *and* $n_D(T_i)$ *as follows. If* $q_i = A = (a, a+1)$ *then* $n_A(T_i) + 1 = n_D(T_i)$*. If* $q_i = B = (b, b)$ *then* $n_A(T_i) = n_D(T_i)$ *and if* $q_i = C = (c+1, c)$ *then* $n_A(T_i) = n_D(T_i) + 1$*.*

Proof: The proof is by induction on i. For $i = 0$, $q_0 = A = (a, a+1)$ and indeed $(n_A(T_0),\ n_D(T_0)) = (1, 2)$ as explained in Example 1.

The induction step follows from the previous description of the recursive formulas for $n_A(T_i)$, and $n_D(T_i)$. For instance, if $q_i = A = (a, a+1)$ then $n_A(T_i) + 1 = n_D(T_i)$. If the $(i+1)^{\text{th}}$ processed symbol satisfies $y_{W-i} = 0$ then $n_A(T_{i+1}) = n_A(T_i) + 1 = n_D(T_i) = n_D(T_{i+1})$ and $q_{i+1} = B = (b, b)$. Thus $\delta(A, 0) = B$. If $y_{W-i} = 1$, then $n_A(T_{i+1}) + 1 = n_A(T_i) + 1 = n_D(T_i) = n_D(T_{i+1})$ and we have that $q_{i+1} = A = (a, a+1)$ and $\delta(A, 1) = A$. Similarly, we can show the correctness of the induction step also for the four other transitions. If $q_i = B = (b, b)$ and $y_{W-i} = 0$ then $n_A(T_{i+1}) = n_A(T_i) + 1 = n_D(T_i) + 1 = n_D(T_{i+1}) + 1$ and $q_{i+1} = C = (c+1, c)$. If $q_i = B = (b, b)$ and $y_{W-i} = 1$ then $n_A(T_{i+1}) + 1 = n_A(T_i) + 1 = n_D(T_i) + 1 = n_D(T_{i+1})$ and $q_{i+1} = A = (a, a+1)$. If $q_i = C = (c+1, c)$ and $y_{W-i} = 0$ then $n_A(T_{i+1}) = n_A(T_i) = n_D(T_i) + 1 = n_D(T_{i+1}) + 1$ and $q_{i+1} = C = (c+1, c)$. If $q_i = C = (c+1, c)$ and $y_{W-i} = 1$ then $n_A(T_{i+1}) = n_A(T_i) = n_D(T_i) + 1 = n_D(T_{i+1})$ and $q_{i+1} = B = (b, b)$. ■

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* $y_W, \ldots, y_1$*. 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 $\delta(C, 1) = B$ while processing the last i symbols in $y_1 \ldots y_W$. We now want to show by induction on i that $n_D(T_i) = n_i + 2$, for $i \in [0, W]$.

First, $n_D(T_0) = 2$ as discussed before and $n_0 = 0$ since the DFA has not yet processed any symbol. For the induction step, we can first see that $n_{i+1} = n_i + 1$ only if the $(i+1)^{\text{th}}$ transition is of the form $\delta(B, 1) = A$ or $\delta(C, 1) = B$ and $n_{i+1} = n_i$ otherwise. Likewise, by the proof of Proposition 4, $n_D(T_{i+1}) = n_D(T_i) + 1$ only if $q_i = B$, and $y_{W-i} = 1$ or $q_i = C$, and $y_{W-i} = 1$, i.e. when $(i+1)^{\text{th}}$ transition is one of the two mentioned above. Thus the equality $n_D(T_{i+1}) = n_{i+1} + 2$ follows from the induction hypothesis $n_D(T_i) = n_i + 2$. By Proposition 1 and the definition of n_y, we can now deduce that $opt_p(R) = n_D(T_W) - 1 = n_W + 2 - 1 = n_W + 1 = n_y + 1$. Finally, by Theorem 1, $opt(R) = opt_p(R) = n_y + 1$. ■

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

$$G(W) = \mathbb{E}_{y:\, 0 \le y \le 2^W - 1}\Big(opt([0, y])\Big) = \frac{1}{2^W} \cdot \sum_{y:\, 0 \le y \le 2^W - 1} opt([0, y]). \tag{3}$$

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

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

Proof: 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.

The transition probabilities are represented in the following 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 $(i,j)^{\text{th}}$ element of P describes the transition probability from the state corresponding to row i to the state corresponding to column j.

$$P = \begin{pmatrix} 0.5 & 0.5 & 0 \\ 0.5 & 0 & 0.5 \\ 0 & 0.5 & 0.5 \end{pmatrix}. \tag{5}$$

Let $r_i = (\Pr(q_i = A), \Pr(q_i = B), \Pr(q_i = C))$. The 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 $\delta(C,1) = B$ among the W performed transitions for each range. Let n_y be the number of these specific transitions while processing y. We think of n_y here as a random variable and present it as the sum of W indicator variables $\{I_{y,i} \mid i \in [1,W]\}$, such that the function $I_{y,i}$ indicates whether the $(i)^{\text{th}}$ transition is one of the two specific transitions. Then we can compute $G(W)$, as presented below.

$$\begin{aligned}
G(W) =& \mathbb{E}_{y:\,0\le y\le 2^W-1}\Big(opt([0,y])\Big) \\
=& \mathbb{E}_{y:\,0\le y\le 2^W-1}\Big(n_y+1\Big) \\
=& 1+\mathbb{E}_{y:\,0\le y\le 2^W-1}\Big(n_y\Big) \\
=& 1+\mathbb{E}_{y:\,0\le y\le 2^W-1}\Big(\sum_{i=1}^{W} I_{y,i}\Big) \\
=& 1+\sum_{i=1}^{W}\mathbb{E}_{y:\,0\le y\le 2^W-1}\Big(I_{y,i}\Big) \\
=& 1+\sum_{i=1}^{W}\Big(\Pr(q_{i-1}=B, y_{W-i+1}=1) \\
&+\Pr(q_{i-1}=C, y_{W-i+1}=1)\Big) \\
=& 1+\frac{1}{2}\cdot\sum_{i=0}^{W-1}\Big(\Pr(q_i=B)+\Pr(q_i=C)\Big) \\
=& 1+\frac{1}{2}\cdot\sum_{i=0}^{W-1}\Big(1-\Pr(q_i=A)\Big) \\
=& 1+\frac{1}{2}\cdot\sum_{i=0}^{W-1}\Big(1-r_i(1)\Big) \\
=& 1+\frac{1}{2}\cdot\sum_{i=0}^{W-1}\Big(1-\big((1,0,0)\cdot P^i\big)_{(1)}\Big) \\
=& 1+\frac{1}{2}\cdot\sum_{i=0}^{W-1}\Big(1-(P^i)_{(1,1)}\Big).
\end{aligned} \tag{6}$$

The matrix P satisfies $(P^{2i-1})_{(1,1)} = (P^{2i})_{(1,1)} = \frac{1}{3} + \frac{2}{3}\cdot\left(\frac{1}{2}\right)^{2i}$. Thus the function $G(W)$ satisfies $G(W) = G(W-2) + \frac{1}{2}\Big((1-(P^{W-2})_{(1,1)}) + (1-(P^{W-1})_{(1,1)})\Big) = G(W-2) + 1 - (P^{W-1})_{(1,1)} = G(W-2) + \frac{2}{3} - \frac{4}{3}\cdot\left(\frac{1}{2}\right)^W$ if W is odd and $G(W) = \frac{1}{2}(G(W-1)+G(W+1))$ if W is even. By solving these recursive equations we directly have the requested formulas for $G(W)$ for the even and odd values of W. ∎

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

By [19]–[21], 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 $G(W) \le \left\lceil \frac{W+1}{2} \right\rceil$. Theorem 3 and its corollary below show that the average encoding length is only about $2/3$ of the worst case.

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

$$\lim_{W\to\infty} \frac{G(W)}{W} = \frac{1}{3}. \tag{7}$$

IV. Analytical Tools For TCAM Expansion Lower Bounds

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, for a range that we need to encode, we find 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* in a certain sense that we make precise later, and as a consequence we deduce that these n pairs cannot be encoded with less than n TCAM entries. Recall that we work over the set of strings of length W, and the width of a TCAM entry is W.

A. Past Results

To prove lower bounds on the TCAM worst-case expansion, we will rely on older analytical tools that have been introduced in [21]. In this section, we review their main definitions and results.

First, let's start with the *hull* of a set of strings.

Definition 4 (Hull)**.** *The hull of n strings $\{a^1,\dots,a^n\}$, where $a^i = a^i_1 \dots a^i_W$, is the smallest cuboid containing $a^1,\dots,a^n$. We denote it by $H(a^1,\dots,a^n)$. Formally,*

$$\begin{aligned} H(a^1,\dots,a^n) = \quad & \{x = x_1\dots x_W \in \{0,1\}^W \mid \\ & \forall j \in [1,W], x_j \in \{a^1_j,\dots,a^n_j\}\} \end{aligned} \tag{8}$$

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

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

Then, the notion of an *alternating path* was defined and used to derive the lower bound in Proposition 6. To simplify the presentation we define the alternating path as well as our relaxation of it to a conflicting set of pairs with respect to ranges, but they both can be defined with respect to a general classification function $\alpha : \{0,1\}^W \to \{accept, deny\}$.

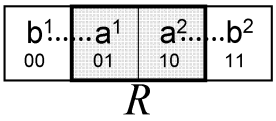


Fig. 3. An illustration of $R = [1, 2] = \{01, 10\}$ from Example 3, and its conflicting set of pairs $B_2 = \{(a^1, b^1), (a^2, b^2)\} = \{(01, 00), (10, 11)\}$ of size $W = 2$. Members of the same pair in B_2 are connected in a dashed line. $a^1, a^2 \in R$ and $b^1, b^2 \notin R$. Likewise, $H(a^1, a^2) \cap \{b^1, b^2\} = \{b^1, b^2\} \neq \emptyset$ and $H(b^1, b^2) \cap \{a^1, a^2\} = \{a^1, a^2\} \neq \emptyset$. Therefore, any encoding of R needs at least $|B_2| = 2$ TCAM entries.

Definition 5 (Alternating Path)**.** *An* alternating path A_n *of size* n *with respect to a range* R *is an ordered set of* $2n-1$ *strings* $A_n = (a^1, \ldots, a^{2n-1})$ *that satisfies the following two conditions:*
(i) Alternation:

$$\{a^1, a^3, \ldots, a^{2n-1}\} \subseteq R, \{a^2, a^4, \ldots, a^{2n-2}\} \cap R = \emptyset. \quad (9)$$

(ii) Hull: *For any* i_1, i_2, i_3 *such that* $1 \leq i_1 < i_2 < i_3 \leq 2n-1$,

$$a^{i_2} \in H(a^{i_1}, a^{i_3}). \quad (10)$$

Proposition 6. *A TCAM encoding of a range with an alternating path of length* n *contains at least* n *entries.*

B. Conflicting Set of Pairs

We now define 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* such that we cannot encode together two points within the range or alternatively two points outside the range from two different pairs.

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 *ordered pairs of strings*

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

that satisfies the following two conditions:
(i) Alternation: *For* $i \in [1, n]$,

$$a^i \in R \text{ and } b^i \notin R. \quad (11)$$

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

$$H(a^{i_1}, a^{i_2}) \cap \{b^{i_1}, b^{i_2}\} \neq \emptyset, \text{ and}$$
$$H(b^{i_1}, b^{i_2}) \cap \{a^{i_1}, a^{i_2}\} \neq \emptyset. \quad (12)$$

Example 3. *Let* $W = 2$*, and consider the range* $R = [1, 2] = \{01, 10\}$ *presented in Fig. 3. Let* $a^1 = 1 = 01$*,* $b^1 = 0 = 00$*,* $a^2 = 2 = 10$ *and* $b^2 = 3 = 11$*. Then* $B_2 = \{(a^1, b^1), (a^2, b^2)\}$ *is a* conflicting set of pairs of size $n = 2$ *because it satisfies the two conditions:*
(i) Alternation: $a^1 \in R$*,* $b^1 \notin R$ *and* $a^2 \in R$*,* $b^2 \notin R$*.*
(ii) Hull: *First,* $b^2 \in H(a^1, a^2)$*, i.e.* $11 \in H(01, 10)$*, because it shares its first bit with* a^2 *and its second bit with* a^1*. Likewise,* $a^2 = 10 \in H(b^1, b^2) = H(00, 11)$*, because it shares its first bit with* b^2 *and its second bit with* b^1*.*

Since the alternation property holds for any pair of elements in a conflicting set of pairs and the hull property holds for any two pairs of elements, 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 removing any pair of elements in the conflicting set of pairs yields a conflicting set of pairs of size* n*.*

The next lemma suggests a lower bound on the range expansion of a range with a conflicting set of pairs.

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

Proof: The proof is *by induction* on n. Let $B_n = \{(a^i, b^i) | i \in [1, n]\}$ denote the conflicting set of pairs and let $\beta_n = \bigcup_{i=1}^{n} \{a^i, b^i\}$ denote the set of elements in the pairs of B_n.

Induction basis: For $n = 1$, let $B_1 = \{(a^1, b^1)\}$. We need at least one TCAM entry to accept a^1.

Induction step: We assume that we cannot encode a classifier function with a conflicting set of pairs of size n in less than n TCAM entries, and want to show it for $n+1$ as well.

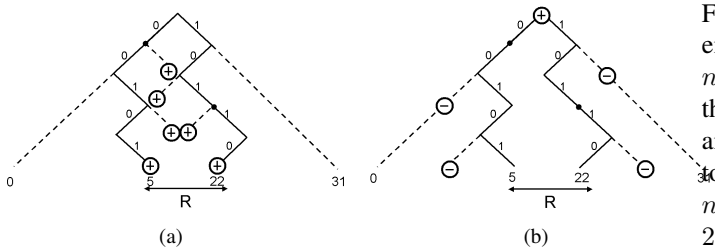
Assume by contradiction that we can encode a classifier function with a conflicting set of pairs $B_{n+1} = \{(a^1, b^1), \ldots, (a^{n+1}, b^{n+1})\}$ of size $n+1$ in less than $n+1$ TCAM entries. Then consider the first TCAM entry $S \rightarrow a$, and distinguish several cases.

(i) If none of the elements of β_{n+1} are in this first TCAM entry, which we denote by $\beta_{n+1} \cap S = \emptyset$, then S does not impact B_{n+1}, and we can actually encode the elements of B_{n+1} in the next (at most) $n-1$ TCAM entries. However, by Observation 5 we can extract from B_{n+1} a conflicting set of pairs of size n, e.g. $\{(a^1, b^1), \ldots, (a^n, b^n)\}$, and by induction we know that it cannot be encoded in $n-1$ TCAM entries.

(ii) If a single element a^i or b^i out of β_{n+1} is in this first TCAM entry, i.e. $\beta_{n+1} \cap S = \{a^i\}$ or $\beta_{n+1} \cap S = \{b^i\}$, then by Observation 5, we can remove from B_{n+1}, the pair of elements (a^i, b^i), and obtain a conflicting set of pairs B_n of size n that does not contain a^i and b^i. But then we need to encode B_n in the next (at most) $n-1$ TCAM entries, because $\beta_n \cap S = \emptyset$, and by induction we know that it is impossible.

(iii) If at least two elements out of β_{n+1} are in this first TCAM entry, i.e. $|\beta_{n+1} \cap S| > 1$, then they all must yield the same action by definition of the TCAM entry. Assume first that $\{a^{i_1}, a^{i_2}\} \subseteq \beta_{n+1} \cap S$, with $i_1 \neq i_2$ and $a^{i_1}, a^{i_1} \in R$. By Definition 6, $\{b^{i_1}, b^{i_2}\} \cap H(a^{i_1}, a^{i_2}) \neq \emptyset$. Therefore, by Proposition 5, at least one of b^{i_1}, b^{i_2} also matches the same TCAM entry, even though they should both yield a different action than a^{i_1} and a^{i_2}. Contradiction again. A similar contradiction occurs if $\{b^{i_1}, b^{i_2}\} \subseteq \beta_{n+1} \cap S$, with $i_1 \neq i_2$ and $b^{i_1}, b^{i_1} \notin R$. ■

Remark: We can in fact prove a stronger version of Lemma 1 saying that even if we give up the element b^n of the pair (a^n, b^n) the lower bound still follows. Consequently it is not hard to see how we can get a conflicting set of n pairs without b^n from an alternating path of length $2n-1$.



(a) (b)

Fig. 4. Two encoding schemes of the range $R = [5, 22] = \{00101, \ldots, 10110\}$ from Example 4. Fig. 4(a) presents the encoding of R itself as a union of several prefix ranges. The six plus signs represents the six TCAM entries in this encoding. Fig. 4(b) demonstrates the alternative encoding of R in which we first encode negatively its complementary R^c and add an additional entry that accepts R itself. Again, the five signs represents the five entries of this encoding. For any W-bit range R, one of these two encodings has at most W entries.

V. Bounds on Worst-Case Expansion

A. General 1-D Ranges

The following result is known from [19]–[21].

Property 6. *For all $W \geq 1$, the maximum range expansion satisfies the following upper-bound:*

$$r(W) \leq r_p(W) = W. \tag{13}$$

In this section we describe a very simple algorithm that encodes a W-bit range with at most W rules. Although this algorithm has the same maximum range expansion as the encoding scheme presented in [19], it is unique in its simplicity. This algorithm either uses entries that accept the range, or a set of entries that deny the complement of the range and an additional entry that accepts everything else.

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

Let $n_0(y), n_1(y)$ be the number of 0s and the number of 1s in $y_{j+1} \ldots y_W$, respectively. Similarly, we define $n_0(z), n_1(z)$ as the number of each of these symbols in $z_{j+1} \ldots z_W$. We can present R as a union of at most $n_0(y) + n_1(z) + 2$ prefix ranges as follows:

$$\begin{aligned} R = \quad & \left(\bigcup_{i \in [j+1,W], y_i = 0} \{y_1 \ldots y_{i-1} 1 (*)^{W-i}\}\right) \bigcup \\ & \left(\bigcup_{i \in [j+1,W], z_i = 1} \{z_1 \ldots z_{i-1} 0 (*)^{W-i}\}\right) \bigcup \{y, z\}. \end{aligned} \tag{14}$$

By adding an action of 'accept' to each of the corresponding prefix TCAM entries, we can encode R in $n_0(y) + n_1(z) + 2$ prefix entries. Likewise, we can represent R^c as a union of $n_1(y) + n_0(z)$ prefix ranges

$$\begin{aligned} R^c = \quad & \left(\bigcup_{i \in [j+1,W], y_i = 1} \{y_1 \ldots y_{i-1} 0 (*)^{W-i}\}\right) \bigcup \\ & \left(\bigcup_{i \in [j+1,W], z_i = 0} \{z_1 \ldots z_{i-1} 1 (*)^{W-i}\}\right). \end{aligned} \tag{15}$$

Then we can encode R by first encoding negatively its complementary R^c by adding an entry of 'deny' to each of these last $n_1(y) + n_0(z)$ entries and adding a last entry of the form $(*)^W \rightarrow$ 'accept' in a total expansion of $n_1(y) + n_0(z) + 1$. For a range R, we denote by $n_a(R), n_d(R)$ the number of entries in these two presented encodings, respectively, such that $n_a(R) = n_0(y) + n_1(z) + 2$ and $n_d(R) = n_1(y) + n_0(z) + 1$. By their definitions, we have that $n_0(y) + n_1(y) = W - j \leq W - 1$ and $n_0(z) + n_1(z) \leq W - 1$ as well. We then have the the total number of entries in these two possible encoding satisfies $n_a(R) + n_d(R) = (n_0(y) + n_1(z) + 2) + (n_1(y) + n_0(z) + 1) \leq 2(W-1) + 3 = 2W + 1$. Thus $\min\{n_a(R), n_d(R)\} \leq W$ and one of the encodings includes at most W entries. We encode R by this encoding.

Example 4. *Let $W = 5$, and consider the range $R = [y, z] = [5, 22] = \{00101, \ldots, 10110\}$. As illustrated in Fig. 4(a), we can encode R itself as union of $n_0(y) + n_1(z) + 2 = 2 + 2 + 2 = 6$ (here $j = 1$) prefix TCAM entries $(01(*)^3 \rightarrow$ 'accept', $0011* \rightarrow$ 'accept', $00101 \rightarrow$ 'accept', $100(*)^2 \rightarrow$ 'accept', $1010* \rightarrow$ 'accept', $10110 \rightarrow$ 'accept'). Likewise, as presented in Fig. 4(b) we can first encode negatively R^c and add a last entry that accepts everything else by $(000(*)^2 \rightarrow$ 'deny', $00100 \rightarrow$ 'deny', $11(*)^3 \rightarrow$ 'deny', $10111 \rightarrow$ 'deny', $(*)^5 \rightarrow$ 'accept'), in $n_1(y) + n_0(z) + 1 = 2 + 2 + 1 = 5$ entries. For this range R, in the second encoding the number of entries satisfies $n_d(R) = 5 \leq W$.*

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 [21] 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 the conclusion of [21].*

Theorem 7. *For all $W \geq 1$, the maximum range expansion satisfies*

$$r(W) = r_p(W) = W. \tag{16}$$

Proof: We show that for all $W \geq 1$, the maximum range expansion satisfies $r(W) \geq W$. It then follows from Property 6 that the bound is tight, i.e. $r(W) = r_p(W) = W$.

To do so, we first assume that W is even, and consider the range $R = \left[\frac{1}{3}\left(2^W - 1\right), 2^{W-1} - 1\right] \bigcup \left[2^{W-1}, \frac{2}{3}\left(2^W - 1\right)\right] = \left[\frac{1}{3}\left(2^W - 1\right), \frac{2}{3}\left(2^W - 1\right)\right] = \{(01)^{\frac{W}{2}}, \ldots, (10)^{\frac{W}{2}}\}$. We show that given R, we can build a conflicting set of pairs of size W. Then, based on Lemma 1, we deduce that R cannot be encoded in less than W TCAM entries.

The construction is as follows. We start by defining $2W$ elements $c^1, c^2, \ldots, c^{2W}$ from which the W pairs of the conflicting set of pairs will be later composed of. We define $c^1 = (01)^{\frac{W}{2}}$, and then obtain $c^2, \ldots, c^{W+1}$ by flipping each time the $(W - (i-1))^{\text{th}}$ bit of c^i to obtain c^{i+1}: by flipping the W^{th} (least significant) bit of c^1, we get $c^2 = (01)^{\frac{W}{2}-1}00$. Then by flipping the $(W-1)^{\text{th}}$ bit of c^2, we get $c^3 = (01)^{\frac{W}{2}-1}10$, and likewise until $c^W = 00(10)^{\frac{W}{2}-1}$ and $c^{W+1} = (10)^{\frac{W}{2}}$. We then continue to obtain $c^{W+2}, \ldots, c^{2W}$ in a similar way, such that c^{i+1} is given by flipping the $(2W - (i-1))^{\text{th}}$ bit of c^i, for $i \in [W+1, 2W-1]$. We can see that $R = \left[\frac{1}{3}\left(2^W - 1\right), \frac{2}{3}\left(2^W - 1\right)\right] = \{(01)^{\frac{W}{2}}, \ldots, (10)^{\frac{W}{2}}\} = [c^1, c^{W+1}]$. We remind that in comparing two W-bit binary strings c^i and c^j using the lexicographic order, $c^i < c^j$ iff there exists some most significant different bit k such that their first

$k-1$ bits are equal, and the k^{th} bit of c^i is 0 while the k^{th} bit of c^j is 1. We can now observe that these $2W$ elements have 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 $i \in [1, W+1]$, c^i has the same first $W-(i-1)$ bits as $c^1 = (01)^{\frac{W}{2}}$ and the same last $i-1$ bits as $c^{W+1} = (10)^{\frac{W}{2}}$. For $i \in [W+1, 2W]$, c^i has the same first $2W-(i-1)$ bits as c^{W+1} and the same last $i-(W+1)$ bits as $c^1 = (01)^{\frac{W}{2}}$.

(iii) For $i \in [2, W]$, the most significant bit in which c^i and c^1 differ is the $(W-(i-2))^{\text{th}}$ 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. From the same reason, by comparing c^i and c^{W+1}, we have that also 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 two required conditions. The alternation property follows directly from *(iii)*.

We would like to show now that B_W satisfies also the hull property. We first consider two elements a^{i_1}, a^{i_2} for $1 \leq i_1 < i_2 \leq W$. If $i_1, i_2 \in [1, \frac{W}{2}]$ or $i_1, i_2 \in [\frac{W}{2}+1, W]$, 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 $i_1 \in [1, \frac{W}{2}]$ and $i_2 \in [\frac{W}{2}+1, W]$, let $i = i_1$ and $j = i_2 - \frac{W}{2}$. We then have $a^{i_1} = a^i = (01)^{\frac{W}{2}-(i-1)}(10)^{(i-1)}$ and $a^{i_2} = a^{j+\frac{W}{2}} = (10)^{\frac{W}{2}-(j-1)}(01)^{(j-1)}$. We distinguish two possible subcases. If $i \geq j$, a^{i_1}, a^{i_2} differ in their first $W-2(i-1)$ first bits. Since a^{i_1}, b^{i_1} differ only in their $(W-2(i-1))^{\text{th}}$ bit, we have that $b^{i_1} \in H(a^{i_1}, a^{i_2})$. From 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 $i_1, i_2 \in [1, \frac{W}{2}]$ or $i_1, i_2 \in [\frac{W}{2}+1, W]$, then $a^{i_2} \in H(b^{i_1}, b^{i_2})$ since a^{i_2}, b^{i_2} differ in only one bit on which a^{i_2} and b^{i_1} agree. If $i_1 \in [1, \frac{W}{2}]$ and $i_2 \in [\frac{W}{2}+1, W]$, we use again the notations i, j as defined above. Here $b^{i_1} = b^i = (01)^{\frac{W}{2}-i}00(10)^{(i-1)}$ and $b^{i_2} = b^{j+\frac{W}{2}} = (10)^{\frac{W}{2}-j}11(01)^{(j-1)}$. 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 $(W-(2j-2))^{\text{th}}$ bit, we have that $a^{i_2} \in H(b^{i_1}, b^{i_2})$. From the same reason, if $i < j$ then $a^{i_1} \in H(b^{i_1}, b^{i_2})$.

Then, we can observe that B_W is indeed a conflicting set of pairs. It has exactly W pairs of elements, thus it is a conflicting set of pairs of size W. Finally, based on Lemma 1, we deduce that R cannot be encoded in less than W TCAM entries.

Likewise, if W is odd, we consider the range $R = \left[\frac{1}{3}\left(2^{W-1}-1\right), \frac{4}{3}\left(2^{W-1}-1\right)\right] = \{0(01)^{\frac{W-1}{2}}, \ldots, (10)^{\frac{W-1}{2}}0\}$. We define $c^1 = 0(01)^{\frac{W-1}{2}}$, and then obtain $(c^1, \ldots, c^W)$ by flipping each time the $(W-(i-1))^{\text{th}}$ bit of c^i to obtain c^{i+1}. For instance, $c^2 = 0(01)^{\frac{W-1}{2}-1}00$ and $c^W = 0(10)^{\frac{W-1}{2}}$. We then continue to obtain $(c^{W+1}, \ldots, c^{2W})$ such that for $i \in [W+1, 2W]$, c^i is given by flipping the first bit of c^{i-W}. By defining again, for $i \in [1, W]$, $a^i = c^{2i-1}$ and $b^i = c^{2i}$, we can show that $B_W = \{(a^1, b^1), \ldots, (a^W, b^W)\}$ is a conflicting set of pairs of size W. Thus R cannot be encoded in less than W TCAM entries. ■

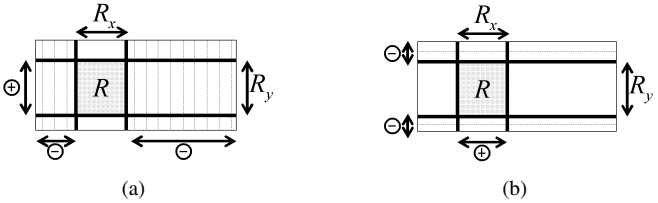


Fig. 5. Two-dimensional range $R = R_x \times R_y$. Fig. 5(a) presents the encoding of the two-dimensional range R by negatively encoding the complementary of R_x and then encoding positively R_y itself. Fig. 5(b) demonstrates the alternative encoding of R: first encoding negatively R_y's complementary and then encoding positively R_x itself.

Example 5. *Let $W = 6$, and consider the range $R = [21, 42] = \{(01)^{\frac{W}{2}}, \ldots, (10)^{\frac{W}{2}}\}$. Let $(a^1, b^1) = (010101, 010100)$, $(a^2, b^2) = (010110, 010010)$, $(a^3, b^3) = (011010, 001010)$. Likewise, let $(a^4, b^4) = (101010, 101011)$, $(a^5, b^5) = (101001, 101101)$, $(a^6, b^6) = (100101, 110101)$. Then, $B_W = \{(a^1, b^1), \ldots, (a^W, b^W)\}$ is a conflicting set of pairs of size W. Thus the range R cannot be encoded in less than $W = 6$ TCAM entries.*

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.

Lemma 2. *A two-dimensional classification rule R has a worst-case expansion of*

$$r^2(W) \leq 2W. \tag{17}$$

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

First, as illustrated in Fig. 5(a), to encode R^2 we can negatively encode the complementary of R_x in $n_d(R_x)-1$ entries while setting the symbols corresponding to the second field to be $(*)^W$ in all entries (here an additional entry that accepts the R_x itself is not required). Then we encode positively R_y itself again with $(*)^W$ in the first field in $n_a(R_y)$. This encodes R in $n_d(R_x)-1+n_a(R_y)$ entries. Alternatively, as demonstrated

in Fig. 5(b), we can first encode negatively $(R_y)^c$ and then R_x itself in $n_d(R_y) - 1 + n_a(R_x)$. As explained in Section V, $n_a(R_x) + n_d(R_x), n_a(R_y) + n_d(R_y) \leq 2W + 1$. We now deduce that $n_d(R_x) - 1 + n_a(R_y) + n_d(R_y) - 1 + n_a(R_x) = n_a(R_x) + n_d(R_x) + n_a(R_y) + n_d(R_y) - 2 \leq 4W$. Thus $\min\{n_d(R_x) - 1 + n_a(R_y), n_d(R_y) - 1 + n_a(R_x)\} \leq 2W$. Finally, we encode R^2 by the encoding that achieves the minimum, and the bound on the expansion is satisfied. ∎

We now show that our suggested encoding scheme for two-dimensional ranges has the optimal worst-case TCAM expansion. To do so, we present a particular two-dimensional range, denoted as 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.

We first generalize Definition 4 for pairs of strings. Consider n pairs of strings $(a^1, b^1), \ldots, (a^n, b^n)$, then the hull of $\{(a^1, b^1) \ldots, (a^n, b^n)\}$ is the cartesian product of the single-dimensional hulls $H((a^1, b^1) \ldots, (a^n, b^n)) = H(a^1 \ldots, a^n) \times H(b^1 \ldots, b^n) = \{(a, b) \mid a \in H(a^1 \ldots, a^n), b \in H(b^1 \ldots, b^n)\}$.

From Proposition 5, it follows that the single-dimensional hull is transitive, that is $H(a^1, \ldots, a^n) = H(H(a^1, \ldots, a^n))$. From this, it easily follows that the multidimensional hull is also transitive.

The following definition and proposition are needed to prove the lower bound on the worst-case expansion of two-dimensional ranges. For a bit value b_i let $\overline{b}_i$ be the bit value $1 - b_i$. We define the function $\Psi : [0, 2^W - 1]^2 \to [0, 2^W - 1]^2$ mapping a pair of strings $(x, y) = (x_1 \ldots x_W, y_1 \ldots y_W)$ to a pair of strings as follows:

$$\Psi((x, y)) = \Psi((x_1 \ldots x_W, y_1 \ldots y_W)) = (y_1 \ldots y_W, \overline{x}_1 \ldots \overline{x}_W). \tag{18}$$

$\Psi((x, y))$ is an injective function and $\Psi^{-1}((x, y)) = (\overline{y}_1 \ldots \overline{y}_W, x_1 \ldots x_W)$.

Proposition 7. *For any pairs of strings b^1, b^2, b^3,*

$$b^3 \in H(b^1, b^2) \Leftrightarrow \Psi(b^3) \in H(\Psi(b^1), \Psi(b^2)). \tag{19}$$

Proof: Let $b^i = (x_1^i \ldots x_W^i, y_1^i \ldots y_W^i)$ for $i \in [1, 3]$. Then, $\Psi(b^i) = (y_1^i \ldots y_W^i, \overline{x}_1^i \ldots \overline{x}_W^i)$. On the one hand, if $b^3 \in H(b^1, b^2)$ then $\forall j \in [1, W], x_j^3 \in \{x_j^1, x_j^2\} \wedge y_j^3 \in \{y_j^1, y_j^2\}$. We immediately also have that $\forall j \in [1, W], \overline{x}_j^3 \in \{\overline{x}_j^1, \overline{x}_j^2\}$ and as a consequence $\Psi(b^3) \in H(\Psi(b^1), \Psi(b^2))$. For the second direction, we can also see that $\Psi^4((x, y)) = \Psi^4((x_1 \ldots x_W, y_1 \ldots y_W)) = \Psi^3((y_1 \ldots y_W, \overline{x}_1 \ldots \overline{x}_W)) = \Psi^2((\overline{x}_1 \ldots \overline{x}_W, \overline{y}_1 \ldots \overline{y}_W)) = \Psi^1((\overline{y}_1 \ldots \overline{y}_W, x_1 \ldots x_W)) = (x_1 \ldots x_W, y_1 \ldots y_W) = (x, y)$. Thus if $\Psi(b^3) \in H(\Psi(b^1), \Psi(b^2))$, we use the previous claim three times and observe that $b^3 = \Psi^4(b^3) \in H(\Psi^4(b^1), \Psi^4(b^2)) = H(b^1, b^2)$. ∎

We can now prove the main result of this section. This result shows the optimality of the suggested encoding scheme for two-dimensional ranges.

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

$$r^2(W) \geq 2W. \tag{20}$$

Proof: We provide the proof for even values of W. We consider the range $R^2 = R \times R = \left[\frac{1}{3}\left(2^W - 1\right), \frac{2}{3}\left(2^W - 1\right)\right] \times \left[\frac{1}{3}\left(2^W - 1\right), \frac{2}{3}\left(2^W - 1\right)\right]$. The projection of R^2 on each dimension is the hard-to-encode range $R = \left[\frac{1}{3}\left(2^W - 1\right), \frac{2}{3}\left(2^W - 1\right)\right]$ from which we can build a conflicting set of pairs of size W, as described in Section V. We also reuse the definitions of $a^1, b^1, \ldots, a^W, b^W$ defined there and remind that $B_W = \{(a^1, b^1), \ldots, (a^W, b^W)\}$ is a conflicting set of pairs of size W as shown in Section V.

We define $4W$ pairs of strings as follows. For $i \in [1, \frac{W}{2}]$, $c^i = (a^i, a^1)$ and $d^i = (b^i, a^1)$. For $i \in [\frac{W}{2} + 1, W]$, $c^i = (a^1, a^i)$ and $d^i = (a^1, b^i)$. Next, for $i \in [W + 1, \frac{3W}{2}]$, $c^i = (a^{i - \frac{W}{2}}, a^{\frac{W}{2}+1})$ and $d^i = (b^{i - \frac{W}{2}}, a^{\frac{W}{2}+1})$. Finally, for $i \in [\frac{3W}{2}+1, 2W]$, $c^i = (a^{\frac{W}{2}+1}, a^{i - \frac{3W}{2}})$ and $d^i = (a^{\frac{W}{2}+1}, b^{i - \frac{3W}{2}})$. For simplicity, we denote these four sets of W indices by S_i for $i \in [1, 4]$ such that $S_i = [(i-1) \cdot \frac{W}{2} + 1, i \cdot \frac{W}{2}]$. We would like to show now that $D_W = \{(c^1, d^1), \ldots, (c^{2W}, d^{2W})\}$ is a conflicting set of pairs of size $2W$.

We start with the *alternation* property and remind that since $B_W = \{(a^1, b^1), \ldots, (a^W, b^W)\}$ is a conflicting set of pairs, we have that $\forall i \in [1, W]$, $a^i \in R, b^i \notin R$. Since a pair of strings (x, y) satisfies $(x, y) \in R^2 = R \times R$ iff $x \in R$ and $y \in R$, we can observe directly from their definitions that $\forall i \in [1, 2W]$, $c^i \in R^2, d^i \notin R^2$.

Next, we examine the *hull* property of D_W. We again use the fact that $B_W = \{(a^1, b^1), \ldots, (a^W, b^W)\}$ itself is a conflicting set of pairs.

We want to show that for any i_1, i_2 such that $1 \leq i_1 < i_2 \leq 2W$, $H(c^{i_1}, c^{i_2}) \cap \{d^{i_1}, d^{i_2}\} \neq \emptyset \wedge H(d^{i_1}, d^{i_2}) \cap \{c^{i_1}, c^{i_2}\} \neq \emptyset$. We consider several cases regarding the membership of i_1, i_2 to the four sets of indices S_1, S_2, S_3 and S_4.

If $i_1, i_2 \in S_1$ then $c^{i_1} = (a^{i_1}, a^1)$ and $c^{i_2} = (a^{i_2}, a^1)$. It is known that $H(a^{i_1}, a^{i_2}) \cap \{b^{i_1}, b^{i_2}\} \neq \emptyset$. If $b^{i_1} \in H(a^{i_1}, a^{i_2})$ then by the generalization of Definition 4, $d^{i_1} = (b^{i_1}, a^1) \in H((a^{i_1}, a^1), (a^{i_2}, a^1)) = H(c^{i_1}, c^{i_2})$. Likewise, if $b^{i_2} \in H(a^{i_1}, a^{i_2})$ then $d^{i_2} = (b^{i_2}, a^1) \in H((a^{i_1}, a^1), (a^{i_2}, a^1)) = H(c^{i_1}, c^{i_2})$. Thus $H(c^{i_1}, c^{i_2}) \cap \{d^{i_1}, d^{i_2}\} \neq \emptyset$. Further, since $H(b^{i_1}, b^{i_2}) \cap \{a^{i_1}, a^{i_2}\} \neq \emptyset$, we have that either $a^{i_1} \in H(b^{i_1}, b^{i_2})$ and $c^{i_1} = (a^{i_1}, a^1) \in H((b^{i_1}, a^1), (b^{i_2}, a^1)) = H(d^{i_1}, d^{i_2})$ or $a^{i_2} \in H(b^{i_1}, b^{i_2})$ and $c^{i_2} = (a^{i_2}, a^1) \in H((b^{i_1}, a^1), (b^{i_2}, a^1)) = H(d^{i_1}, d^{i_2})$. Therefore, $H(d^{i_1}, d^{i_2}) \cap \{c^{i_1}, c^{i_2}\} \neq \emptyset$.

If $i_1 \in S_1, i_2 \in S_2$ then $c^{i_1} = (a^{i_1}, a^1)$ and $c^{i_2} = (a^1, a^{i_2})$. First, $(a^1, a^1) \in H(c^{i_1}, c^{i_2})$ since it shares its first W bits with c^{i_2} and its last W bits with c^{i_1}. Second, $b^{i_2} \in H(a^1, a^{i_2})$ and $d^{i_2} = (a^1, b^{i_2}) \in H((a^1, a^1), (a^1, a^{i_2}))$. Last, based on the transitivity property of the hull function, we observe that $d^{i_2} \in H(c^{i_1}, c^{i_2})$. Likewise, we can see that $a^{i_1} \in H(b^{i_1}, b^{i_2})$ and $c^{i_1} \in H(d^{i_1}, d^{i_2})$.

If $i_1 \in S_1, i_2 \in S_3$ then $c^{i_1} = (a^{i_1}, a^1)$ and $c^{i_2} = (a^{i_2 - \frac{W}{2}}, a^{\frac{W}{2}+1})$. Clearly, $a^1, a^{\frac{W}{2}+1} \in H(a^1, a^{\frac{W}{2}+1})$. Thus based again on the generalization of Definition 4, if $b^{i_1} \in H(a^{i_1}, a^{i_2 - \frac{W}{2}})$ then $d^{i_1} = (b^{i_1}, a^1) \in H((a^{i_1}, a^1), (a^{i_2 - \frac{W}{2}}, a^{\frac{W}{2}+1})) = H(c^{i_1}, c^{i_2})$. If $b^{i_2 - \frac{W}{2}} \in H(a^{i_1}, a^{i_2 - \frac{W}{2}})$ then $d^{i_2} = (b^{i_2 - \frac{W}{2}}, a^{\frac{W}{2}+1}) \in H((a^{i_1}, a^1), (a^{i_2 - \frac{W}{2}}, a^{\frac{W}{2}+1})) = H(c^{i_1}, c^{i_2})$. Thus we have

again that $H(c^{i_1}, c^{i_2}) \cap \{d^{i_1}, d^{i_2}\} \neq \emptyset$. Using the same arguments we can show that $H(d^{i_1}, d^{i_2}) \cap \{c^{i_1}, c^{i_2}\} \neq \emptyset$.

We would like now to generalize these cases and show the existence of the *hull* property also for arbitrary i_1, i_2. To do so, we use Proposition 7. We first observe that for $i \in [1, 2W]$, $\Psi(c^i) = c^{i+\frac{W}{2}}$ and $\Psi(d^i) = d^{i+\frac{W}{2}}$ where the indices summing up is calculated modulu $2W$. For instance, for $i \in [1, \frac{W}{2}]$, $c^i = (a^i, a^1), d^i = (b^i, a^1)$ and $c^{i+\frac{W}{2}} = (a^1, a^{i+\frac{W}{2}})$ and $d^{i+\frac{W}{2}} = (a^1, b^{i+\frac{W}{2}})$. By the definition of a^i, b^i for $i \in [1, W]$, $a^{i+\frac{W}{2}}$ (and $b^{i+\frac{W}{2}}$) is achieved by flipping all the bits of a^i (b^i). Thus we indeed have that $\Psi(c^i) = c^{i+\frac{W}{2}}$ and $\Psi(d^i) = d^{i+\frac{W}{2}}$.

If $i_1 \in S_1, i_2 \in S_4$, then $(i_1 + \frac{W}{2}) \in S_2, (i_2 + \frac{W}{2}) \in S_1$. Then, as proved earlier $d^{i_1+\frac{W}{2}} \in H(c^{i_1+\frac{W}{2}}, c^{i_2+\frac{W}{2}})$ and $c^{i_2+\frac{W}{2}} \in H(d^{i_1+\frac{W}{2}}, d^{i_2+\frac{W}{2}})$. Based on Proposition 7, we have that $d^{i_1} = \Psi^{-1}(d^{i_1+\frac{W}{2}}) \in H(\Psi^{-1}(c^{i_1+\frac{W}{2}}), \Psi^{-1}(c^{i_2+\frac{W}{2}})) = H(c^{i_1}, c^{i_2})$ and similarly $c^{i_2} \in H(d^{i_1}, d^{i_2})$.

More generally, if $i_1 \in S_i, i_2 \in S_j$ for $1 \leq i \leq j \leq 4$ then $(i_1 - \frac{(i-1)\cdot W}{2}) \in S_1, (i_2 - \frac{(i-1)\cdot W}{2}) \in S_{j-(i-1)}$. As shown in one of the four subcases above according to the identity of the set $S_{j-(i-1)}$, $H(\Psi^{-(i-1)}(c^{i_1}), \Psi^{-(i-1)}(c^{i_2})) \cap \{\Psi^{-(i-1)}(d^{i_1}), \Psi^{-(i-1)}(d^{i_2})\} = H(c^{i_1-\frac{(i-1)\cdot W}{2}}, c^{i_2-\frac{(i-1)\cdot W}{2}}) \cap \{d^{i_1-\frac{(i-1)\cdot W}{2}}, d^{i_2-\frac{(i-1)\cdot W}{2}}\} \neq \emptyset$. By applying Proposition 7 $(i-1)$ times, we have that $H(c^{i_1}, c^{i_2}) \cap \{d^{i_1}, d^{i_2}\} \neq \emptyset$. Similar claims show that $H(d^{i_1}, d^{i_2}) \cap \{c^{i_1}, c^{i_2}\} \neq \emptyset$.

Finally, we observe that $D_W = \{(c^1, d^1), \ldots, (c^{2W}, d^{2W})\}$ is a conflicting set of pairs of size $2W$. Thus by Lemma 1 the two-dimensional range R^2 cannot be encoded in less than $2W$ TCAM entries. ∎

It follows from Lemma 2 that the bound is tight, i.e.

$$r^2(W) = 2W. \tag{21}$$

Theorem 8. *The worst-case expansion of a two-dimensional classification rule satisfies,*

$$r^2(W) = r_p^2(W) = 2W. \tag{22}$$

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. ∎

Example 6. *For $W = 2$, consider the two-dimensional range $R^2 = R \times R = [1,2] \times [1,2]$. For the range $R = [1,2]$, $B_W = \{(a^1, b^1), (a^2, b^2)\} = \{(01, 00), (10, 11)\}$ is a conflicting set of pairs. Based on the construction above we define $4W$ pairs of strings as illustrated in Fig. 6. $c^1 = (a^1, a^1)$, $d^1 = (b^1, a^1)$, $c^2 = (a^1, a^2)$, $d^2 = (a^1, b^2)$. Likewise, $c^3 = (a^2, a^2)$, $d^3 = (b^2, a^2)$, $c^4 = (a^2, a^1)$, $d^4 = (a^2, b^1)$. Then $D_2 = \{(c^1, d^1), (c^2, d^2), (c^3, d^3), (c^4, d^4)\}$ is a conflicting set of pairs of size $2W$ and the range R^2 cannot be encoded in less than $2W = 4$ TCAM entries.*

C. Extremal 2-D Ranges

As in the case of one-dimensional ranges, the upper bound for general two-dimensional ranges from Theorem 8 can be improved when only extremal ranges are considered.

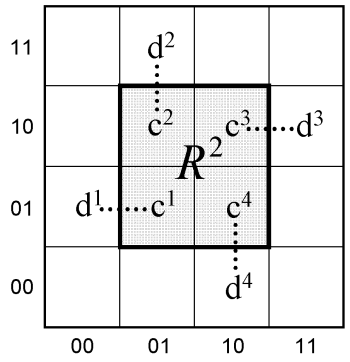


Fig. 6. A conflicting set of pairs $\{(c^1, d^1), (c^2, d^2), (c^3, d^3), (c^4, d^4)\}$ of size $2W = 4$ for the range $R^2 = [1,2] \times [1,2]$. We prove that any encoding of R^2 needs at least 4 TCAM entries.

Lemma 4. *The worst-case expansion of a two-dimensional extremal classification rule R satisfies*

$$r^{e,2}(W) \leq r_p^{e,2}(W) \leq W + 1. \tag{23}$$

Proof: We again consider two possible encodings for the two-dimensional extremal range $R^2 = R_x \times R_y$ as presented in the proof of Lemma 2. Since R_x, R_y are both extremal ranges, we can show that $n_a(R_x) + n_d(R_x), n_a(R_y) + n_d(R_y) \leq W + 2$. Then $n_d(R_x) - 1 + n_a(R_y) + n_d(R_y) - 1 + n_a(R_x) \leq 2W + 2$ and $\min\{n_d(R_x) - 1 + n_a(R_y), n_d(R_y) - 1 + n_a(R_x)\} \leq W + 1$. The range R^2 can be encoded in the smaller encoding among the two. ∎

We would also like to suggest lower bounds on the worst-case expansion of these special range. To do so, we first generalize Proposition 6 to the case of multidimensional strings.

Lemma 5. *Consider d range fields of W bits, and define an alternating path with an* alternation *property and a hull property (generalization of Definition 4). This alternating path is composed of concatenated strings of $d \cdot W$ bits. Then, a d-dimensional classifier function with an alternating path of size n cannot be encoded in less than n TCAM entries.*

Proof: The proof is similar to the proof of Proposition 6 presented in [21] and is given by induction on W. We first show that the encoding of a single string in an alternating path of size $n = 1$ requires one TCAM entry. By the same considerations, given such an alternating path A_n of size n, we cannot encode in one TCAM entry more than one point from the alternating path. Given the first TCAM entry, we can produce as a subset of A_n a new alternating path of size $(n-1)$ such that none of its points are defined in this entry. Thus by the induction hypothesis, at least $n-1$ additional entries are required to complete the encoding of A_n. ∎

Lemma 6. *The worst-case expansion of a two-dimensional extremal classification rule R satisfies*

$$r_p^{e,2}(W) \geq r^{e,2}(W) \geq 2 \cdot \left\lceil \frac{W+1}{2} \right\rceil - 1. \tag{24}$$

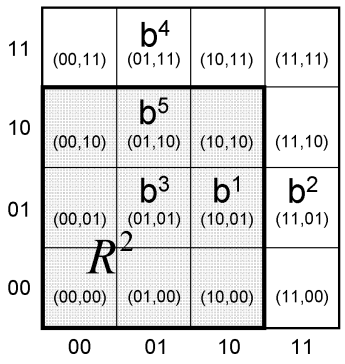


Fig. 7. An illustration of $R^2 = [0,2] \times [0,2]$ from Example 7, and its alternating path $B_{W+1} = (b^1, \ldots, b^5) = ((10,01),(11,01),(01,01),(01,11),(01,10))$ of size $W+1=3$. We prove that any encoding of R^2 requires at least $W+1=3$ TCAM entries.

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

Proof: We first provide the proof for even values of W. The proof uses the one-dimensional range $R = [0, \frac{2}{3}\left(2^W - 1\right)] = \{(0)^W, \ldots, (10)^{\frac{W}{2}}\}$. We start by defining $a^1 = (10)^{\frac{W}{2}}$ and obtaining $(a^2, \ldots, a^{W+1})$ by flipping each time the $(W-(i-1))^{\text{th}}$ bit of a^i to have a^{i+1} for $i \in [1, W]$. Then, as shown in [21], $A_n = (a^1, \ldots, a^{W+1})$ is an alternating path of size $n = \frac{W}{2} + 1$. We now consider the two-dimensional range $R^2 = R \times R$ and build an L-shaped alternating path in the two-dimensional space composed of the two one-dimensional alternating paths joined together. We remind that $(x,y) \in R^2$ if $x, y \in R$. We define the alternating path $B_{2n-1} = (b^1, \ldots, b^{W+1}, \ldots, b^{2W+1})$ of size $2n-1 = 2 \cdot (\frac{W}{2}+1) - 1 = W+1$ in which each element is composed of a pair of strings from the original one-dimensional alternating path. For $i \in [1, W+1]$, we define $b^i = (a^i, a^{W+1})$ and for $i \in [W+1, 2W+1]$, $b^i = (a^{W+1}, a^{2W+2-i})$.

Based on the fact that A_n is an alternating path, we now show that B_{2n-1} satisfies the two required conditions:

(i) Alternation: First, $b^1 = (a^1, a^{W+1}) \in R^2$ since $a^1, a^{W+1} \in R$. Next, for $i \in [1, \frac{W}{2}]$, $b^{2i} = (a^{2i}, a^{W+1}) \notin R^2$ since $a^{2i} \notin R$ and $b^{2i+1} = (a^{2i+1}, a^{W+1}) \in R^2$. Last, for $i \in [\frac{W}{2}+1, W]$, $b^{2i} = (a^{W+1}, a^{2W+2-2i}) \notin R^2$ since $a^{2W+2-2i} \notin R$ and $b^{2i+1} = (a^{W+1}, a^{2W+1-2i}) \in R^2$.

(ii) Hull: We want to show that for any i_1, i_2, i_3 such that $1 \leq i_1 < i_2 < i_3 \leq 2W+1$, $b^{i_2} \in H(b^{i_1}, b^{i_3})$ and consider several cases.

(ii.a) If $i_1 < i_2 < i_3 \leq W+1$, then $b^{i_2} = (a^{i_2}, a^{W+1}) \in H((a^{i_1}, a^{W+1}), (a^{i_3}, a^{W+1})) = H(b^{i_1}, b^{i_3})$ since $a^{i_2} \in H(a^{i_1}, a^{i_3})$.

(ii.b) If $W+1 \leq i_1 < i_2 < i_3 \leq 2W+1$, then $b^{i_2} = (a^{W+1}, a^{2W+2-i_2}) \in H((a^{W+1}, a^{2W+2-i_1}), (a^{W+1}, a^{2W+2-i_3})) = H(b^{i_1}, b^{i_3})$ since $a^{2W+2-i_2} \in H(a^{2W+2-i_1}, a^{2W+2-i_3})$.

(ii.c) If $i_1 < i_2 \leq W+1 \leq i_3 \leq 2W+1$, then $b^{i_2} = (a^{i_2}, a^{W+1}) \in H((a^{i_1}, a^{W+1}), (a^{W+1}, a^{2W+2-i_3})) = H(b^{i_1}, b^{i_3})$, since $a^{i_2} \in H(a^{i_1}, a^{W+1})$ and clearly $a^{W+1} \in H(a^{W+1}, a^{2W+2-i_3})$.

(ii.d) If $i_1 \leq W+1 \leq i_2 < i_3 \leq 2W+1$, $b^{i_2} = (a^{W+1}, a^{2W+2-i_2}) \in H((a^{i_1}, a^{W+1}), (a^{W+1}, a^{2W+2-i_3})) = H(b^{i_1}, b^{i_3})$, since $a^{2W+2-i_2} \in H(a^{W+1}, a^{2W+2-i_3})$ and clearly $a^{W+1} \in H(a^{i_1}, a^{W+1})$.

We now deduce that B_{2n-1} is an alternating path of size $2n-1 = 2 \cdot (\frac{W}{2}+1) - 1 = W+1$ and apply Lemma 5 to have the requested result.

If W is even, we first define the one-dimensional range $R = \{0, \frac{2}{3}\left(2^{W-1}-1\right)\} = \{(0)^W, \ldots, (10)^{\frac{W-1}{2}}\}$ and then consider the two-dimensional range $R^2 = R \times R$. This is the same range we considered for the even value of $W-1$. We again build the L-shaped alternating path of size $(W-1)+1 = W$ in the two-dimensional space as defined above and deduce that the two-dimensional range R^2 cannot be encoded in less than $(W-1)+1 = W$ TCAM entries. ∎

Based on Lemma 4 and Lemma 6 we can now observer the following result.

Theorem 9. *The worst-case expansion of a two-dimensional extremal classification rule satisfies*

$$2 \cdot \left\lceil \frac{W+1}{2} \right\rceil - 1 \leq r^{e,2}(W) \leq r_p^{e,2}(W) \leq W+1. \quad (25)$$

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.

It clearly follows from Lemma 4 and Lemma 6 that the bounds are tight, i.e.

$$r^{e,2}(W) = r_p^{e,2}(W) = W+1 \text{ if } W \text{ is even, and} \quad (26)$$

$$W \leq r^{e,2}(W) \leq r_p^{e,2}(W) \leq W+1 \text{ if } W \text{ is odd.} \quad (27)$$

Example 7. *Consider the two-dimensional range $R^2 = [0,2] \times [0,2]$ with $W = 2$. As illustrated in Fig. 7, $B_{W+1} = ((10,01),(11,01),(01,01),(01,11),(01,10))$ is an alternating path of size $W+1=3$. Thus any encoding of R^2 requires at least $W+1=3$ TCAM entries.*

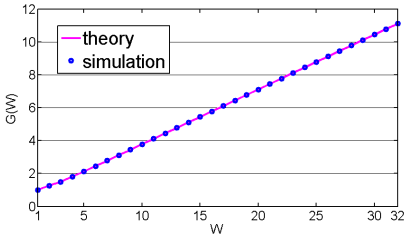
VI. Experimental Results

A. One-Dimensional Extremal Ranges

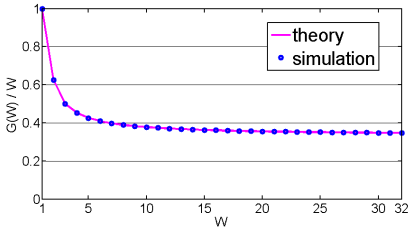
We conduct simulations to examine the results of the average range expansion for extremal ranges presented in Section III-D. Fig. 8(a) presents the function $G(W)$ for $W \in [1, 32]$. For each value of W, the average expansion is calculated based on 2^W extremal ranges. We can see that the simulated 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 $[0,2], [0,5], [0,6], [0,7]$ requires 2 entries.

Next, Fig. 8(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 Theorem 4. For instance, for $W = 16, G(W)/W \approx 0.3611$ and for $W = 32, G(W)/W \approx 0.3472$.

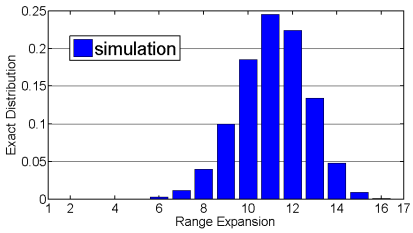
Last, Fig. 8(c) presents the distribution of the extremal range expansion for $W = 32$. The minimal expansion is of course 1 and the maximal expansion is $\lceil \frac{W+1}{2} \rceil = 17$, both with negligible probability. For instance, among the 2^{32} extremal ranges, only 33 extremal ranges of the form $[0, 2^i - 1]$ for $i \in [0, 32]$ can be encoded in only one TCAM entry.



(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 Theorem 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. 8. Simulations of extremal range expansion

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

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 V (with an improved worst-case expansion of $2W$) in comparison with other well-known encoding schemes such as the the Binary Prefix encoding [18], the SRGE encoding [9] and the external encoding for two-dimensional ranges from [19]. The first two schemes use the cartesian product of encodings of the one-dimensional ranges and have a quadratic worst-case expansion of $(2W-2)^2$ and $(2W-4)^2$. The third scheme with both denying and accepting entries first encodes a two-dimensional range $[r_1,r_2]\times[r_3,r_4]\subseteq[0,2^W-1]\times[0,2^W-1]$ by first encoding negatively the four regions $([0,r_1-1]\times[0,2^W-1],[r_2+1,2^W-1]\times[0,2^W-1],[0,2^W-1]\times[0,r_3-1],[0,2^W-1]\times[r_4+1,2^W-1])$ and has a linear worst-case expansion of $2\cdot(2W-2)+1=4W-3$.

For $W\in[4,8]$, we examine all two-dimensional ranges in $[0,2^W-1]\times[0,2^W-1]$. Since each two-dimensional range is defined by two one-dimensional ranges of the form $[r_1,r_2]$ s.t. $0\le r_1<r_2\le(2^W-1)$ or $0\le r_1=r_2\le(2^W-1)$, the total number of two-dimensional ranges is $\left(\binom{2^W}{2}+\binom{2^W}{1}\right)^2$.

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-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).

Last, we compare the average range expansion for the two-dimensional ranges in these real-life files. The average obtained range expansions in the Binary Prefix encoding, the external encoding from [21] and our suggested encoding were 47.18, 20.09 and 12.56, respectively. This shows an improvement of 73.4% and 37.5% of the suggested scheme when compared to the other two encodings.

D. Effectiveness on Real-life Packet Classifiers

Fig. 9(a) presents the total expansion of the two-dimensional ranges in twelve artificial classifiers generated by the ClassBench benchmark tool [26] 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 $2W$) in comparison with Binary Prefix encoding [18] 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 and 78.2% and 78.0%, respectively. Likewise, for the real-life files, the improvement is 73.4% in comparison with Binary Prefix.

Fig. 9(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 as in Binary Prefix encoding. Thus, the total improvement is

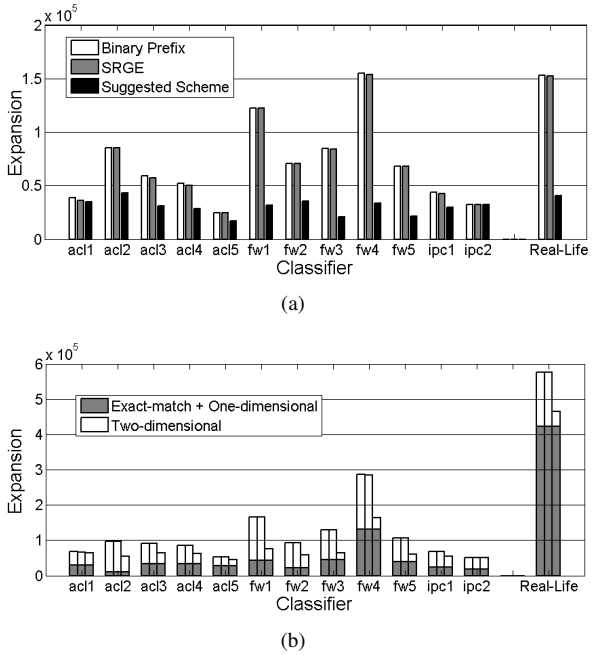


Fig. 9. Effectiveness of the suggested encoding scheme and the suggested joint TCAM architecture (illustrated in Fig. 1) on twelve artificial classifiers generated by ClassBench benchmark tool and on a real-life database. For each classifier, the two left bars present the expansion of Binary Prefix and of SRGE. 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.

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.

VII. 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.

Two intersecting directions for future work encompass extending our optimal analysis to general range rules, and studying the optimal encoding of an arbitrary set of rules.

VIII. Acknowledgment

We would also like to thank Noam Nisan, Danny Raz, Alex Shpiner and Aran Bergman for their helpful participation and suggestions. We would 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 an Intel research grant on Heterogeneous Computing and by the Hasso Plattner Center for Scalable Computing. Ori Rottenstreich is the Google Europe Fellow in Computer Networking 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] A. X. Liu, C. R. Meiners, and Y. Zhou, "All-match based complete redundancy removal for packet classifiers in TCAMs," in *IEEE Infocom Mini-Conference*, 2008.
[14] 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.
[15] 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.
[16] 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.
[17] E. Spitznagel, D. E. Taylor, and J. S. Turner, "Packet classification using extended TCAMs," in *ICNP*, 2003.
[18] V. Srinivasan, G. Varghese, S. Suri, and M. Waldvogel, "Fast and scalable layer four switching," in *ACM SIGCOMM*, 1998.
[19] O. Rottenstreich and I. Keslassy, "Worst-case TCAM rule expansion," in *IEEE Infocom Mini-Conference*, 2010.
[20] R. Cohen and D. Raz, "Simple efficient TCAM based range classification," in *IEEE Infocom Mini-Conference*, 2010.
[21] O. Rottenstreich and I. Keslassy, "On the code length of TCAM coding schemes," in *IEEE ISIT*, 2010.
[22] O. Rottenstreich, R. Cohen, D. Raz, and I. Keslassy, "Exact worst-case TCAM rule expansion," *IEEE Trans. Computers*, 2012.
[23] W. Jiang and V. K. Prasanna, "Scalable packet classification on FPGA," *IEEE Trans. VLSI Syst.*, vol. 20, no. 9, pp. 1668–1680, 2012.
[24] 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.
[25] Y. Ma and S. Banerjee, "A smart pre-classifier to reduce power consumption of TCAMs for multi-dimensional packet classification," in *ACM SIGCOMM*, 2012.
[26] D. E. Taylor and J. S. Turner, "ClassBench: a packet classification benchmark," in *IEEE Infocom*, 2005.