CGFE: Efficient Range Encoding for TCAMs*

Jérôme Graf[†‡] Vitalii Demianiuk[‡] Pavel Chuprikov[†] Yang Feiran[§] Sergey I. Nikolenko[¶] Patrick Eugster[†]
[†]Università della Svizzera italiana [‡]TurbaAI [§]Huawei [¶]Harbour Space University

***Abstract*—High-performance packet classification is essential for a wide range of fundamental network functions, including access control, firewalls, and advanced programmable network applications. Ternary content addressable memory (TCAM) is heavily used for packet classification due to its impressive performance but it is size-limited, expensive, and power-intensive. Since TCAM requires special encoding for ranges in packet classifiers, minimizing the encoding size is essential. Classical methods such as DIRPE and SRGE work well for different types of ranges and have different limitations. We present the chunked Gray fence encoding (CGFE), a novel encoding that combines the advantages of DIRPE's fence encoding and Gray code reflectivity in SRGE, achieving the best of both worlds. CGFE reduces the number of TCAM entries needed for range-based packet classifiers, improving TCAM efficiency and lowering energy consumption. We prove that CGFE uses the same or smaller number of TCAM entries than both DIRPE and SRGE for *every* possible range, reducing the number of ternary strings up to $2\times$ in theory and, as we show in a comprehensive practical evaluation, by 40.8% and 9.3% on average, respectively, for rules with two 16-bit range fields.**

I. Introduction

The ever-growing demands of network functionality necessitate high-performance packet classification across network appliances, to optimize network performance and functionality. This includes traditional security tools such as firewalls [1] and access control lists [2], as well as programmable switches and sketches [3] for more dynamic and flexible network control and traffic management. Especially modern architectures such as software-defined networking and network function virtualization rely on efficient packet classification [4, 5, 6, 7].

Packet classifier rules target specific fields in packet headers such as IP addresses or port numbers and define actions to perform with a packet, e.g., forwarding, dropping, redirecting, logging, if the header *matches* one of the rules [8]. Some rule fields (e.g, port numbers) define matches by *ranges* of values.

Ternary content addressable memory (TCAM) has evolved to become the industry standard for packet classification, widely adopted in networking equipment and network processing units [9]. TCAM is composed of fixed-width ternary entries, each being a string consisting of 0, 1, and $\star$ ("don't care") bits. Given a binary string, a TCAM can match it against all ternary entries in parallel. Thus, TCAM can provide high throughput unparalleled by software-based solutions [10]. Despite the ubiquity of TCAM, it remains a challenge to efficiently express *range-based rules* in them.

Solutions for this problem can in general be divided into *database-independent* and *database-dependent* approaches. The former approaches [11, 12, 13] encode all ranges using the same technique, thus allowing for fast hot updates but suffer from the so-called *range expansion* problem: a single range-based rule often needs to be encoded to multiple TCAM entries, and the number of entries grows exponentially with the number of range fields. In contrast, database-dependent approaches [14, 15] often achieve more compact representations but suffer from limitations in update speed and scalability for a large number of different ranges [16]. Since modern network applications often deal with large and dynamic rule sets that require near real-time processing [17, 18, 19] and contain many different ranges, in this work we concentrate on the database-independent approaches. A common database-independent approach is *prefix encoding* [12] that converts a range into individual prefixes, each represented by a separate TCAM entry. In the worst-case scenario, a single rule specifying a 16-bit range field would require $2 \cdot 16 - 2 = 30$ TCAM entries; if the same rule specifies two 16-bit range fields, prefix encoding would use up to $30 \cdot 30 = 900$ entries.

While TCAM offers superior speed, it is limited in size and has high energy consumption [20, 21, 22, 23, 24, 25]. As shown, naïve encoding approaches for range-based rules can significantly inflate the number of TCAM entries required, limiting the system's ability to handle complex or large packet classifiers. Efficient range encoding techniques can significantly reduce the number of TCAM entries needed to represent a set of range-based rules. This reduction translates into two key benefits. First, it allows for a larger and more diverse classifier to be accommodated within limited TCAM space, enhancing the overall flexibility and functionality of a packet classification system. Second, by reducing the number of TCAM entries one can achieve a substantial decrease in energy consumption, a critical factor for energy-efficient network devices in the modern data-intensive environment.

We thus introduce a novel approach, chunked Gray fence encoding (CGFE), designed to address the challenges of efficient range encoding in TCAMs for packet classification. By combining the strengths of existing methods, database-independent range pre-encoding (DIRPE) [11] and short range Gray encoding (SRGE) [13], CGFE offers a comprehensive solution capable of efficiently encoding both short and long

* Work financially supported by Hasler Foundation grant #21021, Swiss National Science Foundation grant #192121, and FIR grant "Safe and modular control plane-aware programming of the network data plane".

ranges. Leveraging DIRPE's fence encoding technique and integrating a generalization of reflective Gray code akin to SRGE, CGFE achieves superior performance in TCAMs resource utilization and energy consumption. Thus, our primary contributions are as follows: (1) *innovative database-independent encoding technique* CGFE that combines the benefits of DIRPE and SRGE and better encodes range-based rules in TCAM; (2) *extensive theoretical analysis*, where we prove various properties of CGFE, e.g., that CGFE provides the same or smaller range encoding (in TCAM entries) than DIRPE and SRGE for **any** possible range, achieving up to $2\times$ improvement in the best case scenario; (3) *comprehensive experimental evaluation*, where we test CGFE on all possible ranges in 16-bit range fields, showing reduced TCAM entry counts across various scenarios against both DIRPE and SRGE; e.g., CGFE reduces the amount of required TCAM entries for rules with two 16-bit range fields compared to DIRPE and SRGE on average by 18.5% and 34.6% for short ranges (≤ 64 values per range) and 9.3% and 40.8% for all possible ranges respectively. By minimizing the number of TCAM entries, CGFE reduces both resource utilization and also energy consumption, addressing two critical challenges in TCAM-based packet classification.

The paper is structured as follows. § II provides background on packet classifiers, range-based rule encodings, and existing range encoding techniques. § III introduces CGFE, detailing its algorithm and advantages over existing methods, including formal analysis. § IV evaluates CGFE through comprehensive simulations, analyzing its resource utilization and energy consumption w.r.t. DIRPE and SRGE. § V positions CGFE in the broader field of range encoding for TCAM-based packet classification. Finally, § VI summarizes our key findings and outlines potential future research directions.

II. Background

This section establishes the core terminology employed throughout the paper in line with the terminology used in the field [11, 13, 26], defines the range encoding problem, and summarizes commonly used range encoding methods.

A. Packet Classification Definition

A packet *header* $\mathcal{H} = (H_1, ..., H_k)$ is a tuple of k *header fields*, where each field H_i is a w_i-bit number. Classifiers match headers to determine the respective actions for incoming packets. A *classifier* $\mathcal{K} = \{R_1, ..., R_N\}$ is an ordered set of rules, where each *rule* $R_j = (\mathcal{F}_j, A_j)$ consists of a filter $\mathcal{F}_j = (F_1, ..., F_k)$ and the associated *action* A_j. In the most general case, each *filter field* F_i in a filter $\mathcal{F}_j$ is a range corresponding to a packet header field H_i.

We say that filter field F_i *matches* a header field H_i if H_i belongs to the range F_i; rule $R_j = (\mathcal{F}_j, A_j)$ *matches* a header $\mathcal{H}$ if every filter field F_i in the filter $\mathcal{F}_j$ matches the corresponding field H_i in $\mathcal{H}$. The goal of classification is to find the action of the first matching rule in a classifier $\mathcal{K}$ (see Tab. II). If there is no such rule, a default action is returned.

TABLE I: Main definitions and notation

Notation	Definition	Meaning
$\mathcal{H}$	$\mathcal{H} = (H_1, ..., H_k)$	Packet header
H		Packet header field
k	$k = \lvert\mathcal{H}\rvert$	Header size
w	$w = \lvert H\rvert = \lvert F\rvert$	Header/filter field width
$\mathcal{K}$	$\mathcal{K} = \{R_1, ..., R_N\}$	Classifier
R	$R = (\mathcal{F}, A)$	Classifier rule
$\mathcal{F}$	$\mathcal{F} = (F_1, ..., F_k)$	Filter
F		Filter field
A		Action
$\mathcal{E}$		Encoding method
$\mathfrak{F}(x)$	$0^{2^w-x-1}1^x$	Fence encoding of value x
$\mathfrak{F}([s,e])$	$0^{2^w-e-1}\star^{e-s}1^s$	Fence encoding of range $[s, e]$
c		Chunk size (number of bits)
$\text{MSC}_c(x)$		Most significant chunk of x
$\text{TC}_c(x)$		Tail chunks of x
$r \circ \text{TC}_c(E)$		Reflected extension of E over r
$x \diamond E$		Prepend value x to encoding E

TABLE II: A sample 3-field (k=3) classifier with 3 rules R_1–R_3, each field consisting of 4-bit ranges (w=4). For a packet header $\mathcal{H} = (6, 14, 5)$ the classification result is the action A_2 since $\mathcal{H}$ is matched by the rule R_2.

$\mathcal{K}$	F_1	F_2	F_3	Action
R_1	[0,5]	[3,13]	[14,15]	A_1
R_2	[6,9]	[14,15]	[0,7]	A_2
R_3	[10,15]	[0,2]	[8,13]	A_3

B. Range Encoding Problem

Ternary content addressable memory (TCAM) was introduced for efficient packet classification as an extension of the classical content addressable memory (CAM). For a given binary string B, CAM finds the memory address of the first binary string coinciding with B using multiple parallel searches on the hardware level. TCAM is an extension of CAM allowing it to search over ternary strings consisting of symbols `0`, `1`, or $\star$ ("don't care"). A ternary string T *matches* a binary string B if T coincides with B in all non-$\star$ positions; TCAM finds the first ternary string in memory matching B. Below, we use the notion of a ternary string and TCAM entry interchangeably. Searches in both CAM and TCAM are extremely efficient as all memory cells are compared simultaneously with a given search key.

To perform packet classification on $\mathcal{K}$ in TCAM, every filter $\mathcal{F}_j$ in $\mathcal{K}$ is encoded into a set of ternary bit strings and put in TCAM following $\mathcal{K}$'s rule order, and every packet header $\mathcal{H}$ is encoded into a binary string. Then, by locating the first matching ternary string representing a filter in TCAM, the classification process can determine the action of a first matching rule. Formally, a *filter encoding method* should encode every filter $\mathcal{F}_j$ into a set of ternary strings $\mathcal{E}(\mathcal{F}_j)$ and every packet header $\mathcal{H}$ into the binary string $\mathcal{E}(\mathcal{H})$ such that $\mathcal{H}$ is matched by rule R_j iff at least one ternary string in $\mathcal{E}(\mathcal{F}_j)$ matches $\mathcal{E}(\mathcal{H})$. E.g., $\mathcal{F}_2$ of rule R_2 in Tab. II can be encoded into the following set of ternary strings $\mathcal{E}(\mathcal{F}_2) =$

$\{011\star\ 111\star\ 0\star\star\star,\ \ 100\star\ 111\star\ 0\star\star\star\}$. Packet header $\mathcal{H} = (6, 14, 5)$ encoded into binary string $\mathcal{E}(\mathcal{H}) = 0110\ 1110\ 0101$ matches $011\star\ 111\star\ 0\star\star\star$ from $\mathcal{E}(\mathcal{F}_2)$.

Minimizing the encoding of a classifier is crucial for efficient TCAM utilization. The complex circuitry required for high-speed searching on ternary strings significantly limits the scalability and size of TCAM and also comes with very high energy consumption compared to other types of memory, e.g., CAM or static random access memory (SRAM). A large number of ternary strings per filter translates to a larger footprint within TCAM, limiting the size of classifiers that can be stored and increasing energy consumption and search time, finally impacting the overall classification performance.

Database-independent filter encoding methods are designed to encode any filter with given field widths efficiently. Such a method $\mathcal{E}$ typically encodes fields independently of each other: for a packet header $\mathcal{H}$, $\mathcal{E}$ encodes each field in $\mathcal{H}$ independently by a binary string and then concatenates these strings in the field order. For a filter $\mathcal{F}_j$ the encoding $\mathcal{E}$ encodes every F_i in $\mathcal{F}_j$ independently into a set of ternary strings; then, $\mathcal{E}$ constructs all combinations of ternary strings encoding different fields in $\mathcal{F}_j$; for every combination, $\mathcal{E}$ concatenates strings from this combination into a separate element in the encoding $\mathcal{E}(\mathcal{F}_j)$. E.g., for the encoding of filter $\mathcal{F}_2$ (cf. rule R_2 of Tab. II), first all fields get encoded, i.e., $\mathcal{E}(F_1) = \{011\star, 100\star\}$, $\mathcal{E}(F_2) = \{111\star\}$, and $\mathcal{E}(F_3) = \{0\star\star\star\}$, and then every combination is concatenated to create the encoding of the filter $\mathcal{E}(\mathcal{F}_2) = \{011\star\ 111\star\ 0\star\star\star, 100\star\ 111\star\ 0\star\star\star\}$.

In the worst case, the number of ternary strings representing the filter $\mathcal{F}_j$ grows exponentially with the number of fields in $\mathcal{F}_j$. Thus, the efficiency of a single range encoding can have a large influence on encoding size via exponential blowup. For example, each field in filter $\mathcal{F}_3$ in Tab. II can be encoded into 2 ternary strings, leading to a total of 8 TCAM entries for this single rule; by halving the encoding size of any field in R_3 from 2 ternary strings to 1, we halve the encoding size of $\mathcal{F}_3$ from 8 to 4. Therefore, for the remainder of this work, we focus on encoding individual fields represented by ranges in a classifier. A *range encoding method* encodes a range $r = [s, e]$ (representing a field in a filter) into a set of ternary strings $\mathcal{E}(r)$, and a value a (representing a header field) into a binary string $\mathcal{E}(a)$ such that $\mathcal{E}(a)$ is matched by at least one of the ternary strings in $\mathcal{E}(r)$ if and only if $s \leq a \leq e$. We define the *length* $|r|$ of range $r = [s, e]$ as the number of values in r, i.e., $|r| = e - s + 1$. In the rest of the paper, we assume that bits in binary/ternary strings encoding values/ranges start from the most significant bit.

C. Encoding Techniques for Natural Numbers and Ranges

1) Prefix expansion: this is a straightforward approach to represent range-based rules in TCAMs that encodes values of header fields by their direct binary representations on w bits. Prefix expansion decomposes a range into multiple subranges such that each subrange can be encoded by a single ternary bit string containing $\star$ only after all 0s and 1s; we call such ternary strings *prefixes*. Fig. 1 shows the prefix tree for bit

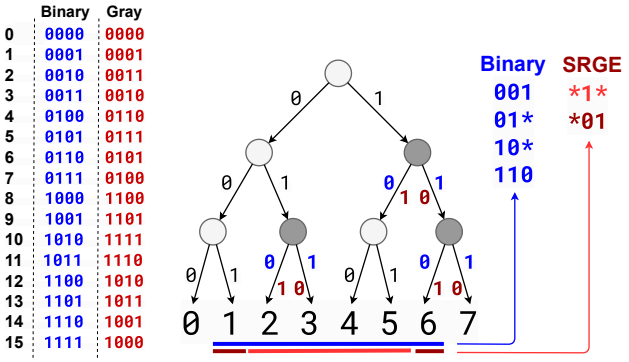


Fig. 1: Binary and Gray encodings for $w = 4$ and sample binary prefix and SRGE encodings for $[1, 6]$ for $w = 3$.

length $w = 3$ and the table of binary encodings for $w = 4$. For $w = 3$, prefix encoding decomposes the range $[1, 6]$ into 4 subranges $\{[1, 1], [2, 3], [4, 5], [6, 6]\}$ and encodes these subranges by 4 prefixes $\{001, 01\star, 10\star, 110\}$ respectively. This also showcases the worst-case expansion for a range rule with prefix expansion, which is $2w - 2$. To achieve a more compact representation, different ways of encoding ranges and consequently different ways of encoding natural numbers need to be explored. Two such techniques are particularly relevant to our new approach: Gray codes and fence encoding.

2) Gray codes: Gray code, or reflected binary code, encodes numbers by binary strings such that encodings of any two adjacent numbers x and $x + 1$ differ in one bit. This minimal change between sequential values is crucial in certain applications such as error correction. The Gray code of a number x can be obtained from its binary representation as follows: (1) find positions of all 1's in the binary representation of x; (2) for each position t found on step (1), flip the value of the $(t+1)$th bit in x. E.g., for the value 12 on $w = 4$ bits, the Gray code 1010 differs from the binary representation 1100 in the second and third positions. Fig. 1 shows the conversion of decimal numbers to binary and Gray codes for $w = 4$ and highlights the differing bits in the prefix tree for $w = 3$.

The main advantage of Gray codes for range encoding is their reflectivity as described in SRGE [13] and RENE [16]. In particular, we can look at SRGE as the extension of prefix expansion using the reflectivity of Gray codes to reduce the number of TCAM entries. The Gray code of value $2^{w-1} - x$ on w bits always coincides with the Gray code of value $2^{w-1} + x - 1$ in all positions but the first. E.g., for $w = 4$ bits, the Gray code of $6 = 2^3 - 2$ is $\mathbf{0}101$ and the Gray code of $9 = 2^3 + 2 - 1$ is $\mathbf{1}101$. Thus, to encode a range $[2^{w-1} - x, 2^{w-1} + x - 1]$, it is sufficient for SRGE to encode $[2^{w-1} - x, 2^{w-1}]$, and then replace the first bit in every ternary string in the encoding by $\star$. For $w = 3$, SRGE encodings of ranges $[1, 3]$ and $[1, 6]$ are $\{\mathbf{0}01, \mathbf{0}1\star\}$ and $\{\star01, \star1\star\}$, respectively. Observe that for $r = [1, 6]$ on $w = 3$ bits, prefix expansion produces two times more ternary strings than SRGE. Also SRGE encodes every range of length 2 with one ternary string, while prefix expansion produces two ternary strings in 50% of cases.

Similarly to prefix expansion, SRGE decomposes a range

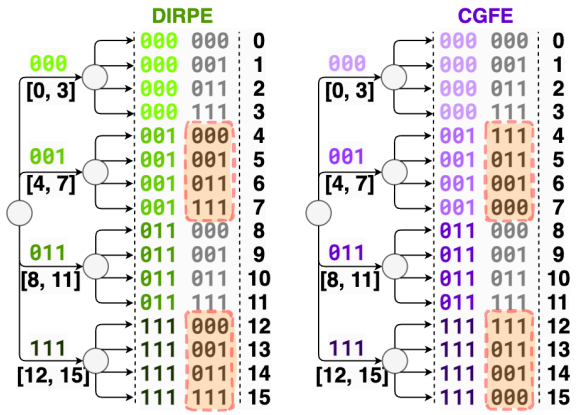


Fig. 2: DIRPE and CGFE encodings of values on $w = 4$ bits split into $c = 2$-bit chunks; shaded regions show differences.

r into subranges corresponding to single prefixes. But, unlike prefix expansion, SRGE can encode multiple subranges by a single ternary string using the reflectivity of Gray codes. E.g., prefix expansion decomposes range $[1, 13]$ on $w = 4$ bits into 5 subranges $[1, 1], [2, 3], [4, 7], [8, 11], [12, 13]$, then encodes each subrange by a single prefix; SRGE encodes pairs of subranges ($[2, 3], [12, 13]$) and ($[4, 7], [8, 11]$) by one ternary string per pair due to Gray code reflectivity.

3) Fence encoding: Fence encoding is a special type of unary coding proposed by Lakshminarayanan et al. [11]. It encodes a w-bit number x as a binary string $0^{2^w-x-1}1^x$, i.e., zero-to-many 0s followed by zero-to-many 1s. In the following, the fence encoding of a value x is denoted $\mathfrak{F}(x)$. E.g., $\mathfrak{F}(4) = \texttt{0001111}$ for 3-bit value 4. This allows a single ternary string to represent any w-bit range $[s, e]$ by $0^{2^w-e-1}\star^{e-s}1^s$. E.g., the 3-bit range $[3, 5]$ is encoded as $\texttt{00}\star\star\texttt{111}$. However, a significant drawback of this approach is the exponential growth in the length of encoded values with the number of bits w. This makes it highly impractical: e.g., 65536 bits are needed for a $w = 16$-bit port number.

Lakshminarayanan et al. [11] addressed this issue with database-independent range pre-encoding (DIRPE) that uses a chunking approach inspired by multibit tries known from IP address lookup, trading some efficiency for practicality. To balance the pros and cons of fence encoding, DIRPE splits a binary string of length w into chunks, encodes each chunk separately by fence encoding, and concatenates the results. E.g., to encode the binary value $x = \texttt{01001110}_2$:

(1) split x into 2-bit chunks, i.e., `01`, `00`, `11`, `10`,
(2) encode each chunk with fence encoding, i.e., `001`, `000`, `111`, `011`, and finally
(3) concatenate all the chunk encodings into the resulting binary string `001000111011`.

The DIRPE encoding of numbers in $w = 4$ bits with a chunk size of $c = 2$ is illustrated in Fig. 2. Similarly to prefix expansion, DIRPE decomposes a range into multiple subranges, encoding each by a single ternary string with the following structure: (1) every chunk in the ternary string is a fence encoding of the range of values for the corresponding chunk in the header field encoding; (2) if a chunk in a ternary string encodes a range with at least two values, the succeeding chunks consist of only $\star$. E.g., for uniform chunk size of $c = 2$ bits, DIRPE encodes subrange $[36, 43]$ on $w = 6$ bits by single ternary string $\{\texttt{011 0}\star\texttt{1 }\star\star\star\}$; first chunk is a fence encoding of value $2 = \texttt{01}_2$, second chunk is a fence encoding 2-bit range $[1, 2]$, the remaining chunk consist of only $\star$.

III. Chunked Gray fence encoding (CGFE)

This section introduces chunked Gray fence encoding (CGFE), our novel approach for efficient range rule encoding in TCAM. CGFE leverages the strengths of the two encoding techniques discussed above: fence encoding and Gray codes.

A. Overview

CGFE bridges the gap between DIRPE and SRGE by transforming fence encoding in a way that gives it the reflectivity property of Gray codes. As a result, CGFE is able to outperform both techniques. While Gray codes excel at efficiently representing short ranges due to their minimal bit difference property, fence encoding offers broader applicability across various range lengths. CGFE gets the best of both worlds, achieving efficiency of to Gray codes for short ranges and maintaining the flexibility of fence encoding for longer ranges. In § III-E we prove that CGFE uses the same or smaller number of TCAM entries than both methods across all ranges.

B. Encoding a Single Number

Similar to DIRPE, CGFE splits a packet field's original binary representation into chunks, encodes each chunk separately with fence encoding, and concatenates the results. Both DIRPE and CGFE work with heterogeneous chunk sizes, but for ease of exposition we assume below that both DIRPE and CGFE split every value x into c-bit chunks and the length of every encoded binary string is divisible by c.

The key difference between DIRPE and CGFE encodings of a value x, the latter denoted as CGFE(x), is the following: if the value of the ith chunk is odd, then CGFE encodes the $(i+1)$th chunk with the c-bit value p, by $\mathfrak{F}(2^c-1-p)$ instead of $\mathfrak{F}(p)$. E.g., CGFE(x) for a 8-bit value $x = \texttt{01001110}_2$ and chunk size $c = 2$ is as follows:

(1) split x into 2-bit chunks, i.e., `01`, `00`, `11`, `10`;
(2) encode each chunk with fence encoding, i.e., `001`, `111`, `111`, `001`; note that the second chunk $\texttt{00}_2$ and the last chunk $\texttt{10}_2$ are encoded by $\mathfrak{F}(3-0) = \mathfrak{F}(3)$ and $\mathfrak{F}(3-2) = \mathfrak{F}(1)$ respectively;
(3) concatenate all chunk encodings into the resulting binary string `001111111001`.

Fig. 2 compares CGFE and DIRPE for $w = 4$ and $c = 2$; the difference is only in the inverted parts that take up half of every second layer of the tree. Constructing the CGFE encoding of a w-bit value takes the same number of logical gates as DIRPE, $O(w \cdot 2^c/c)$ [11], plus $O(w)$ gates to negate chunks succeeding chunks with odd values. Thus, both CGFE and DIRPE require $O(w \cdot 2^c/c)$ gates to encode a w-bit value.

Below, we present CGFE(x) in a recursive form matching the presentation of CGFE range encoding later in the section. We start with some useful notation for chunk manipulation. For a w-bit value x, we define $\text{MSC}_c(x)$ as the *most significant chunk* (of size c), i.e., the value of the chunk containing the c most significant bits of x, and $\text{TC}_c(x)$ as the *tail chunks* (assuming first chunk of size c), i.e., the $(w-c)$-bit value obtained by removing x's c most significant bits. In particular, we have $\text{TC}_c(x) = x - 2^{w-c} \cdot \text{MSC}_c(x)$. For a CGFE encoding E of some w-bit value and c-bit value p, we define $p \diamond E$ as:

$$p \diamond E = \begin{cases} \mathfrak{F}(p), \mathfrak{F}(2^c - 1 - a), \text{TC}(E), & p \text{ is odd}, \\ \mathfrak{F}(p), E & p \text{ is even}. \end{cases}$$

It is not hard to check that CGFE(x) = MSC(x) $\diamond$ CGFE(TC(x)).

Moreover, we can easily generalize this operation to encode ranges. For a CGFE encoding E of some w-bit range and a c-bit value p, we define the *prepend* operation $p \diamond E$ that returns an encoding of a $(w+c)$-bit range formed by replacing each $E_i \in E$, $\text{MSC}(E_i) = \mathfrak{F}([a, b])$, with E'_i defined as

$$E'_i = \begin{cases} \mathfrak{F}(p), \mathfrak{F}([2^c - 1 - b, 2^c - 1 - a]), \text{TC}(E_i), & p \text{ is odd}, \\ \mathfrak{F}(p), E_i, & p \text{ is even}. \end{cases}$$

Observation 1. *For a CGFE encoding E of an w-bit range $[s, e]$ and a c-bit value p, $p \diamond E$ is an encoding of the $(w+c)$-bit range $[p \cdot 2^w + s, p \cdot 2^w + e]$.*

E.g., for the encoding $E = \{000\ \star 11, 001\ \star\star\star\}$ of the range $[2, 7]$ for $w = 4$ and $c = 2$, $1 \diamond E = \{001\ 111\ \star 11, 001\ 011\ \star\star\star\}$ encodes the 6-bit range $[18, 23]$.

C. Reflectivity of CGFE

The power of CGFE comes from its chunk-based reflectivity, whose essence is that for certain related values their CGFE encodings differ in all but one *chunk*. Reflectivity integrates very well with the fence encoding (§ II-C3) enabling efficient CGFE encoding of ranges, for many values can be covered with a single TCAM entry, as we show below.

To present the CGFE reflectivity property, we extend the tail chunks notation to encodings: for an encoding E, we define $\text{TC}_c(E)$ to be E stripped of the bits corresponding to the most-significant c-bit chunk of *input*, i.e., $2^c - 1$ bits of E for CGFE.

CGFE's main reflective property is captured by Thm. 1.

Theorem 1. *For chunk size c and w-bit values x and y, $\text{TC}_c(CGFE(x)) = \text{TC}_c(CGFE(y))$ if (with $\%$ denoting modulo) either (1) $\text{MSC}_c(x)\%2 = \text{MSC}_c(y)\%2$ and $\text{TC}_c(x) = \text{TC}_c(y)$, or (2) $\text{MSC}_c(x)\%2 \neq \text{MSC}_c(y)\%2$ and $\text{TC}_c(x) = 2^{w-c} - 1 - \text{TC}(y)$.*

Proof: Case (1) is straightforward according to the definition of CGFE. Case (2) is also straightforward but only for values consisting of at most two chunks. For other cases, we prove the theorem via mathematical induction. Note that $\text{TC}_c(x) = 2^{w-c} - 1 - \text{TC}_c(y) = 2^{w-c} - 1 - \text{MSC}_c(\text{TC}_c(y)) \cdot 2^{w-2c} - \text{TC}_c(\text{TC}_c(y)) = (2^c - 1 - \text{MSC}_c(\text{TC}_c(y))) \cdot 2^{w-2c} + (2^{w-2c} - 1 - \text{TC}_c(\text{TC}_c(y)))$. Thus, $\text{MSC}_c(\text{TC}_c(x)) = 2^c - 1 - \text{MSC}_c(\text{TC}_c(y))$ and $\text{TC}_c(\text{TC}_c(x)) = 2^{w-2c} - 1 - \text{TC}_c(\text{TC}_c(y))$.

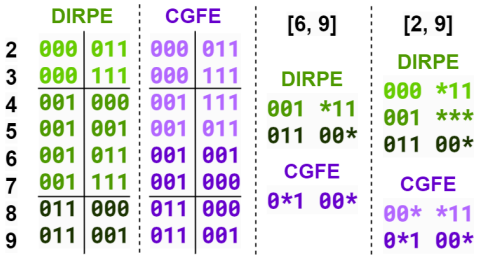


Fig. 3: DIRPE vs CGFE encodings for ranges $[6, 9]$ and $[2, 9]$ on $w = 4$, $c = 2$

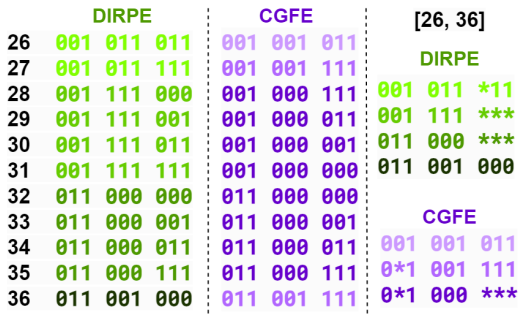


Fig. 4: DIRPE vs CGFE encoding for the range $[26, 36]$ on $w = 6$ bits with $c = 2$.

Since $\text{MSC}_c(x)\%2 \neq \text{MSC}_c(y)\%2$, CGFE either encodes $\text{MSC}_c(\text{TC}_c(x))$ by $\mathfrak{F}(\text{MSC}_c(\text{TC}_c(x)))$ and $\text{MSC}_c(\text{TC}_c(y))$ by $\mathfrak{F}(2^c - 1 - \text{MSC}_c(\text{TC}_c(y))) = \mathfrak{F}(\text{MSC}_c(\text{TC}_c(x)))$ or vice versa. In both cases, the second chunk in CGFE encodings of x and y will be represented by the same bits.

Since $\text{MSC}_c(\text{TC}_c(x)) + \text{MSC}_c(\text{TC}_c(y)) = 2^c - 1$, $\text{MSC}_c(\text{TC}_c(x))\%2 \neq \text{MSC}_c(\text{TC}_c(y))\%2$. Thus, the conditions from the case (2) of the theorem are also satisfied for $\text{TC}_c(x)$ and $\text{TC}_c(y)$. By induction, the theorem is correct for $\text{TC}_c(x)$ and $\text{TC}_c(y)$ implying that bits representing $\text{TC}_c(\text{TC}_c(x))$ and $\text{TC}_c(\text{TC}_c(y))$ in CGFE encodings of $\text{TC}_c(x)$ and $\text{TC}_c(y)$ are the same. Since the first chunk affects only the encoding of the second chunk, CGFE encodings of x and y also coincide at chunks corresponding to $\text{TC}_c(\text{TC}_c(x))$ and $\text{TC}_c(\text{TC}_c(y))$. ■

Thm. 1 can be checked via Fig. 2, e.g., CGFE(7) = 001 000 and CGFE(8) = 011 000 in Fig. 2 differ only in the first chunk since $\text{MSC}(7) = 2, \text{MSC}(8) = 3, \text{TC}(7) + \text{TC}(8) = 3$; similarly, CGFE(5) = 001 011 and CGFE(13) = 111 011 since $\text{MSC}(5) = 1$, $\text{MSC}(13) = 3$ and $\text{TC}(5) = \text{TC}(13) = 1$.

The reflective property reduces the size of range encoding. Take the range $r = [6, 9]$ from Fig. 3, as an example. Following Thm. 1, TC(CGFE(6)) = TC(CGFE(9)) = 001 and TC(CGFE(7)) = TC(CGFE(8)) = 000. Consequently, if an encoding E can be found for the interval $[6, 7]$ we can adapt the first chunk of this encoding (in this case by replacing the first chunk 001 by $\mathfrak{F}([\text{MSC}(6), \text{MSC}(9)]) = 0\star1$) to additionally cover the interval $[8, 9]$.

Formally, to encode the range $[k \cdot 2^{w-c} - x - 1, k \cdot 2^{w-c} + x]$, where $1 \leq k \leq 2^c - 1$, it is sufficient for CGFE to encode the range $[k \cdot 2^{w-c} - x - 1, k \cdot 2^{w-c} - 1]$ as E, remove the

first chunk of every string in E getting $\text{TC}(E)$, and prepend $\mathfrak{F}([k-1,k])$. Note that DIRPE has to encode two ranges: $[k{\cdot}2^{w-c}{-}x{-}1, k{\cdot}2^{w-c}{-}1]$ and $[k{\cdot}2^{w-c}, k{\cdot}2^{w-c}{+}x]$ leading to two times larger encoding than CGFE. If we denote the above operation of prepending $\mathfrak{F}(r)$, for a range r, to E' as $r \circ E'$, the result can be concisely described as $[k-1,k] \circ \text{TC}(E)$. We call $r \circ \text{TC}(E)$ *a reflected extension of* E and will usually denote it as E^{ext}. Reflected extensions are key to CGFE encoding.

CGFE, as DIRPE, uses a chunk-based approach for the encoding of a range r to encode bigger subranges of r by single ternary strings. Similarly to SRGE, CGFE exploits its reflectivity to reuse encoding of some subranges of r to encode values from other subranges of r. For example, DIRPE splits range $r = [2,9]$ in Fig. 3 into three subranges $\{[2,3],[4,7],[8,9]\}$, and then encodes each subrange by a single ternary string; at the same time, CGFE encodes subranges $[2,3]$ and $[8,9]$ by a single ternary string each, and then constructs the encoding of r containing only reflected extensions $[\text{MSC}(2), \text{MSC}(5)] \circ \text{TC}(\text{CGFE}([2,3])) = \texttt{00}\star\ \star\texttt{11}$ and $[\text{MSC}(6), \text{MSC}(9)] \circ \text{TC}(\text{CGFE}([8,9])) = \texttt{0}\star\texttt{1 00}\star$ since these reflected extensions also encode all values in $[4,7]$. Note that CGFE can encode multiple subranges by a single ternary string even if most significant chunks of the values in these subranges differ by more than one. E.g., CGFE in Fig. 4 encodes $[27,27] \cup [36,36]$ as a reflected extension $[\text{MSC}(27), \text{MSC}(36)] \circ \text{TC}(\text{CGFE}([27,27])) = \texttt{0}\star\texttt{1 001 111}$.

D. Range Encoding

We now present the CGFE encoding algorithm (Alg. 1).

To encode a w-bit range $[s,e]$ with $\text{MSC}(s) = \text{MSC}(e)$, we call such a range *local*, CGFE recursively constructs the encoding E of the $(w-c)$-bit range $[\text{TC}(s), \text{TC}(e)]$ and then returns $\text{MSC}(s) \diamond E$ (line 3). Hence, in the following, we assume that $\text{MSC}(s) \neq \text{MSC}(e)$ in a given range $[s,e]$.

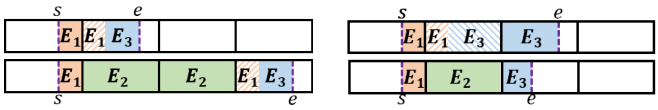
1) Encoding of middle ranges: We say that a w-bit range $[s,e]$ is a *middle* range if $\text{TC}(s) = 0$ and $\text{TC}(e) = 2^{w-c}-1$. CGFE encodes such a range $[s,e]$ by a single ternary string: $[\text{MSC}(s), \text{MSC}(e)] \circ \star^{w-c}$ (line 5).

2) Encoding of top and bottom ranges: We say that a w-bit range $[s,e]$ is a *bottom* range if $\text{TC}(s) = 0$ and $\text{TC}(e) \neq 2^{w-c}-1$. To encode such a range, CGFE splits $[s,e]$ into two ranges $r_1 = [s, x-1]$ and $r_2 = [x,e]$, where $x = \text{MSC}(e) \cdot 2^{w-c}$. CGFE encodes r_1 and r_2 independently: r_1 is a middle range encoded with $[\text{MSC}(s), \text{MSC}(e){-}1] \circ \star^{w-c}$ (line 7); since $\text{MSC}(x) = \text{MSC}(e)$, r_2 is local and can be encoded recursively as $\text{MSC}(e) \diamond \text{CGFE}([0, \text{TC}(e)])$ (line 8). We say that a w-bit range $[s,e]$ is a *top* range if $\text{TC}(s) \neq 0$ and $\text{TC}(e) = 2^{w-c}-1$. The encoding of top ranges is symmetrical to the encoding of bottom ranges. Bottom and top ranges can be encoded with fewer entries if some of their values is known to be already covered by other entries. We call such a reduced encoding *partial* and describe it later in § III-D4.

3) Encoding regular ranges: Now consider a w-bit *regular range* $[s,e]$, where $\text{TC}(s) > 0$, $\text{TC}(e) < 2^{w-c}-1$. It can be represented as a union of local range $r_1 = [s, (\text{MSC}(s)+1) \cdot 2^{w-c}-1]$, middle range $r_2 = [(\text{MSC}(s)+1) \cdot 2^{w-c}, \text{MSC}(e) \cdot 2^{w-c}-1]$ (empty if $\text{MSC}(e) - \text{MSC}(s) = 1$), and local range $r_3 = [\text{MSC}(e){\cdot}2^{w-c}, e]$. Two cases are distinguished by CGFE:

(1) $\text{MSC}(e) - \text{MSC}(s)$ *is odd*: in this case we assume w.l.o.g. that $|r_1| \leq |r_3|$ (the other case is symmetric). First, CGFE encodes r_1 as $E_1 = \text{CGFE}(r_1)$ (line 17) and constructs its a reflected extension $E_1^{\text{ext}} = [\text{MSC}(s), \text{MSC}(e)] \circ \text{TC}(E_1)$ (line 18). Note, E_1^{ext} encodes a range that covers $r' = [\text{MSC}(e) \cdot 2^{w-c}, \text{MSC}(e) \cdot 2^{w-c} + |r_1|]$ that intersects with r_3. If $|r_1| < |r_3|$ the bottom range r_3 is still not fully covered and CGFE constructs its partial encoding, E_3, assuming that first $|r_1|$ values of r_3 are already covered by other entries (cf line 19). If $|r_2| = 0$, no additional encoding is needed. Otherwise, CGFE adds an additional ternary string $E_2 = [\text{MSC}(s){+}1, \text{MSC}(e){-}1] \circ \star^{w-c}$ that encodes r_2 (line 20).

(2) $\text{MSC}(e) - \text{MSC}(s)$ *is even*: in this case, CGFE constructs encodings E_1 and E_3 of r_1 and r_3, respectively, (line 23-24). If $|r_1| + |r_3| \geq 2^{w-c}$, reflected extensions $E_1^{\text{ext}} = [\text{MSC}(s), \text{MSC}(e) - 1] \circ \text{TC}(E_1)$ of E_1 and $E_3^{\text{ext}} = [\text{MSC}(s) + 1, \text{MSC}(e)] \circ \text{TC}(E_3)$ of E_3 encode ranges whose union cover r_2 (line 26-27). Hence in this case, CGFE simply returns $E_1^{\text{ext}} \cup E_3^{\text{ext}}$. Otherwise, CGFE returns $E_1 \cup E_3 \cup [\text{MSC}(s){+}1, \text{MSC}(e){-}1] \circ \star^{w-c}$, where the last ternary string encodes r_2 (line 30-31).



(a) Case: $\text{MSC}(e) - \text{MSC}(s)$ is odd, i.e., in this case 1 or 3 respectively. Note that the reflected extension of E_1 partially covers the bottom range r_3, so E_3 is a partial encoding.

(b) Case: $\text{MSC}(e) - \text{MSC}(s)$ is even, i.e., in this case 2 for both examples. In the lower example, the reflected extensions of E_1 and E_3 are not enough to fully cover the middle chunk so an additional encoding E_2 is needed.

Fig. 5: Illustration of how CGFE encodes regular ranges for a chunk size of $c = 2$ which produces quarters for every chunk. Dashed areas indicate reflected extension.

This encoding of regular ranges by CGFE is illustrated in Fig. 5 for the example of a chunk size of $c = 2$.

4) Partial encoding of bottom and top ranges: Now suppose that the encoding of values in a subrange $r' = [s, e_1 - 1]$ of a given bottom range $r = [s,e]$ is not necessary, that is, we are allowed to encode any range $[s_1, e]$ such that $s_1 < e_1$, choosing s_1 to minimize the encoding size (partial encoding of top range is symmetrical). Such partial encodings are used by CGFE encoding of regular ranges.

If $e_1 > e$, CGFE returns $\varnothing$. If $\text{MSC}(e_1) = \text{MSC}(e)$, the resulting encoding E_1 may not cover any value x with $\text{MSC}(x) < \text{MSC}(e_1) = \text{MSC}(e)$. Hence, in this case CGFE recursively finds the encoding E_1 of a $(w{-}c)$-bit range $[0, \text{TC}(e)]$ under the assumption that encoding values in $[0, \text{TC}(e_1) - 1]$ is not necessary, and then returns $\text{MSC}(e) \diamond E_1$. If $\text{MSC}(e_1) < \text{MSC}(e)$, the given range r can be represented as a union of ranges $r_1 = [0, e_1 - 1]$, $r_2 = [e_1, x-1]$, and $r_3 = [x,e]$, where

Algorithm 1 CGFE encoding with homogeneous chunk size c; we abstract partial encoding in CGFE_PARTIAL (§ III-D4).

Input: a range $r = [s, e]$ where s and e are w-bit values
Output: a set of TCAM entries that covers the range r

function CGFE($[s, e]$)
  if $\text{MSC}(s) = \text{MSC}(e)$ **then** // *local* range
    return $\text{MSC}(s) \diamond \text{CGFE}([\text{TC}(s), \text{TC}(e)])$
  else if $\text{TC}(s) = 0$ and $\text{TC}(e) = 2^{w-c} - 1$ **then**
    return $[\text{MSC}(s), \text{MSC}(e)] \circ \star^{w-c}$ // *middle* range
  else if $\text{TC}(s) = 0$ **then** // *bottom* range
    return $[\text{MSC}(s), \text{MSC}(e) - 1] \circ \star^{w-c}$
      $\cup\, \text{MSC}(e) \diamond \text{CGFE}([0, \text{TC}(e)])$
  else if $\text{TC}(e) = 2^{w-c} - 1$ **then** // *top* range
    // symmetric to *bottom* range
  else // *regular* range
    $r_1 \leftarrow [s, (\text{MSC}(s) + 1) \cdot 2^{w-c} - 1]$
    $r_2 \leftarrow [(\text{MSC}(s) + 1) \cdot 2^{w-c}, \text{MSC}(e) \cdot 2^{w-c} - 1]$
    $r_3 \leftarrow [\text{MSC}(e) \cdot 2^{w-c}, e]$
    if $\text{MSC}(e) - \text{MSC}(s)$ is odd **then**
      // assuming $|r_1| \leq |r_3|$, the other case is symmetric
      $E_1 \leftarrow \text{CGFE}(r_1)$
      $E_1^{\text{ext}} \leftarrow [\text{MSC}(s), \text{MSC}(e)] \circ \text{TC}(E_1)$
      $E_3 \leftarrow \text{CGFE\_PARTIAL}(r_3, |r_1|)$
      $E_2 \leftarrow [\text{MSC}(s) + 1, \text{MSC}(e) - 1] \circ \star^{w-c}$
      return $E_1^{\text{ext}} \cup E_2 \cup E_3$
    else // $\text{MSC}(e) - \text{MSC}(s)$ is even
      $E_1 \leftarrow \text{CGFE}(r_1)$
      $E_3 \leftarrow \text{CGFE}(r_3)$
      if $|r_1| + |r_3| \geq 2^{w-c}$ **then**
        $E_1^{\text{ext}} \leftarrow [\text{MSC}(s), \text{MSC}(e) - 1] \circ \text{TC}(E_1)$
        $E_3^{\text{ext}} \leftarrow [\text{MSC}(s) + 1, \text{MSC}(e)] \circ \text{TC}(E_3)$
        return $E_1^{\text{ext}} \cup E_3^{\text{ext}}$
      else
        $E_{13} \leftarrow \text{CGFE}(r_1) \cup \text{CGFE}(r_3)$
        return $E_{13} \cup [\text{MSC}(s) + 1, \text{MSC}(e) - 1] \circ \star^{w-c}$

$x = \text{MSC}(e) \cdot 2^{w-c}$. Let $E_3 = \text{CGFE}(r_3)$. If $|r_3| \geq |r_2|$, the reflected extension $E_3^{\text{ext}} = [0, \text{MSC}(e)] \circ \text{TC}(E_3)$ of E_3 also covers r_2, so in this case CGFE returns E_3^{ext}. If $|r_3| < |r_2|$, CGFE constructs the encoding of r covering all its values.

E. Provable Properties of CGFE

The following theorems show that the number of ternary strings in CGFE does not exceed the corresponding number for SRGE, DIRPE, and prefix expansion for every range:

Theorem 2. *For every w-bit range r, the number of ternary strings in the CGFE encoding of r, does not exceed the number of ternary strings in the DIRPE encoding of r, under the same chunk size.*

Proof: CGFE, as DIRPE, uses fence encoding and splits binary strings into chunks. Therefore, CGFE maintains DIRPE's efficiency on every range. The reflectivity of CGFE is an additional property that DIRPE lacks, which makes it possible to improve on DIRPE. ■

Corollary 1. *The number of ternary strings in the CGFE encoding of a w-bit range is at most $2\frac{w}{c} - 1$.*

Note that the runtime of the CGFE encoding construction linearly depends on the encoding size.

Theorem 3. *For every w-bit range r, the number of ternary strings in the CGFE encoding of r, does not exceed the number of ternary strings in the SRGE encoding of r.*

Proof: CGFE with $c = 1$ is equivalent to SRGE for any w. Any sequence of chunk sizes can be transformed into chunks of size 1 by repeatedly splitting the first chunk of size $c > 1$ into chunks of size 1 and $c - 1$, e.g., $[1, 3, 2]$ gets transformed into $[1, 1, 1, 1, 1, 1]$ via $[1, 1, 2, 2]$ and $[1, 1, 1, 1, 2]$. The proof (see Appendix) shows by case analysis that for any such split and any range r the size of CGFE encoding of r before the split is at most the size of one after the split. ■

Corollary 2. *For every w-bit range r, the number of ternary strings in the CGFE encoding of r, does not exceed the number of ternary strings in the prefix expansion of r.*

Lemma 1. *CGFE with chunk size $c > 1$, $c = o(w)$, can reduce the number of ternary strings in the encoding of a filter with k range fields on w bits by $O(w^k)$ ternary strings in absolute values and by $2^k - o(1)$ times in relative values compared to DIRPE, SRGE and prefix expansion.*

Proof: To prove the theorem it is sufficient to show it for $k = 1$. Range $[1, 2^w - 2]$ reaches the desired reduction. ■

IV. EVALUATION

We evaluate the efficiency of CGFE in representing packet classifiers with range-based rules. We compare CGFE to state-of-the-art methods w.r.t *encoding size* – the average number of ternary strings required to express ranges/packet classifiers with same properties. We address four research questions:

RQ1: How does the range length influence encoding size of CGFE comparing to other methods?
RQ2: How does the number of range fields in a rule affect the encoding size of classifier for CGFE and other methods?
RQ3: How do chunk sizes affect the differences in encoding sizes of DIRPE and CGFE?
RQ4: How do CGFE's encoding size savings translate to reduced energy consumption per TCAM search operation?

Methodology. We compare CGFE's performance against prefix expansion, DIRPE, and SRGE. We employ synthetic datasets with controlled properties to systematically evaluate CGFE's performance across different range lengths, numbers of range fields in rules, and chunk sizes, to show CGFE's benefits across different scenarios and types of ranges. Our experiments focus on rules containing 1 and 2 range fields with bit-width $w = 16$, aligning with typical applications classifying traffic by source and destination port ranges (experiments with different values for w, omitted for space reasons).We define *expansion reduction* of a range encoding method as a percent reduction of its encoding size compared to the baseline (bigger values correspond to more efficient methods).

RQ1. In line with related work [13], Fig. 6 shows the expansion reduction of different encoding methods, considering prefix expansion as a baseline, for rules having one and two range fields with range length $|r| \leq x$, where x varies from 2 to 2^{16}; DIRPE and CGFE both use a homogeneous

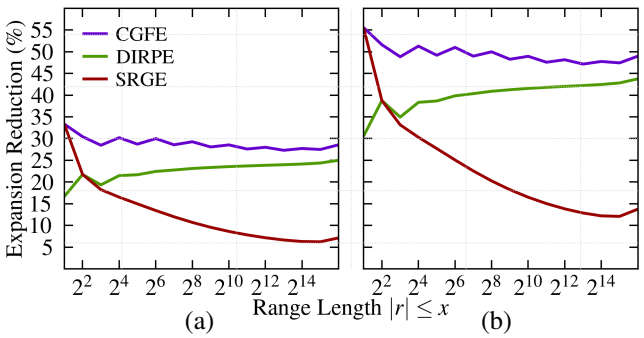


Fig. 6: Average reduction (higher is better) in encoding sizes of different range encoding methods vs prefix expansion w.r.t. range length for rules having (a) 1 and (b) 2 range fields.

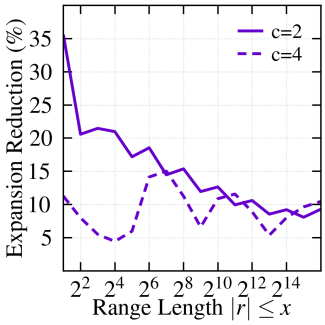


Fig. 7: Avg. reduction in encoding size of CGFE vs DIRPE w.r.t. range length for rules with 2 range fields.

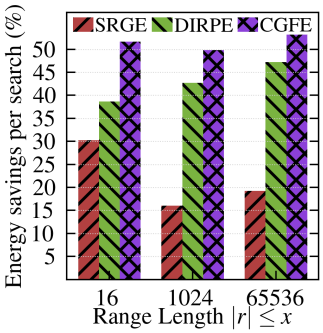


Fig. 8: Energy savings per search (higher is better), in % of prefix expansion for rules with 2 range fields.

chunk size of $c = 2$. While SRGE excels for short ranges, its efficiency drops quickly with longer ranges. DIRPE has a lower initial reduction but remains stable, even improving slightly for longer ranges. CGFE clearly achieves consistently higher reduction than both SRGE and DIRPE across all ranges. In the case of one 16-bit range field in a rule with range length bounded by $x = 64$, CGFE reduces the number of ternary strings in the filter encoding by 30.0%, 19.1%, 14.2% on average compared to prefix encoding, SRGE, and DIRPE, respectively; for all range lengths, the reduction is 28.7%, 22.5%, 4.8% on average, respectively.

RQ2 Comparison of Fig. 6a and Fig. 6b shows how the impact of the range encoding method on the classifier encoding size increases rapidly with the growth of the number of range fields in a rule. For example in Fig. 6, expansion reduction of CGFE on two ranges is at least 1.66× higher than for CGFE on one range for every range length bound x. Moreover, the differences in encoding sizes of CGFE and other range encoding methods are bigger for rules with 2 range fields than for those with 1 range field. In particular, for rules with two 16-bit ranges, CGFE reduces the number of ternary strings in the filter encoding by 49.0%, 40.8%, 9.3% on average compared to prefix encoding, SRGE, and DIRPE, respectively. In the general case, CGFE savings in the encoding size exponentially depend on the number of range fields in a rule.

RQ3 Fig. 7 hones in on expansion reduction of CGFE, with DIRPE as baseline, for rules having two 16-bit ranges of length $|r| \leq x$ and x varying from 2 to 2^{16}. Larger chunks significantly improve efficiency for both methods, but require more extra bits. For instance, encoding two 16-bit fields with eight 2-bit chunks each takes 16 extra bits in total, whereas encoding the same fields with four 4-bit chunks each requires a significantly higher 88 extra bits in total but can lead up to taking only half the ternary strings in the encoding. Fig. 7 shows that CGFE maintains an advantage over DIRPE under the same chunk size with expansion reduction varying from 5 to 35% percent depending on chunk size and range length.

RQ4 Building upon the reduced TCAM requirements, we now analyze the energy consumption per search operation; as discussed earlier, TCAM energy consumption scales directly with the number of entries stored. To confirm that lower TCAM footprint of CGFE translates to significant energy savings, we generate classifiers and determine consumption per search operation with the TCAM power model from [27]. Every classifier consists of 10,000 IPv4 5-tuple rules containing two 16-bit range fields each, with maximum lengths $\leq$ 16, 1024, and 65536. As in TCAM architectures the size of each entry is preconfigured and typically falls into specific values such as 72, 144, 288, or 576 bits [28], we use an entry size of 144 bits with all methods since it is the minimum size needed for an IPv4 5-tuple and yet has enough space for the extra bits of CGFE and DIRPE with chunk size $c = 2$. Fig. 8 shows energy reduction of CGFE, DIRPE and SRGE vs prefix encoding. As expected, CGFE consistently exhibits lowest energy consumption per search. W.r.t. prefix expansion, CGFE reduces energy by 51.7%, 49.8%, and 53.2% for maximum range lengths of 16, 1024, and 65536 respectively.

V. Related Work

Several works explore redesigning TCAM circuits [29, 30, 31, 32, 33] or combining TCAMs with other hardware [2, 34] to improve efficiency; they are promising in terms of resource utilization and reducing energy consumption but are hard to implement on tens of millions of already deployed network devices. Another research direction focuses on compressing the classifier itself [35, 36, 37], looking for an equivalent classifier using less TCAM space; this approach, is hard to extend to dynamic rule updates since re-compressing the entire classifier leads to high update latency. Representations proposed in [10, 26, 38] use structural properties of classifiers, e.g., order-independence of rules, to reduce the number of fields or field width. These representations can be combined with CGFE or any other range encoding method. Other prior works focus on representing range rules with the least TCAM entries. These approaches are compatible with existing rule set compression techniques or future hardware changes and fall into two classes: (1) data-dependent encoding techniques tailor the encoding scheme to a specific rule set and the

ranges inside, potentially achieving higher efficiency [14, 39]; they usually use extra bits as a bitmap, assigning one bit to each range; (2) data-independent approaches such as CGFE are more versatile. Prefix expansion [12] represents ranges as a set of prefixes, leading to a worst-case expansion of $2w-2$ entries for a w-bit field. DIRPE [11] utilizes hierarchical fence encoding to reduce the number of entries but introduces additional bits for fences. SRGE [13] leverages reflective Gray coding, efficient for short ranges without extra bits. However, longer ranges might require more entries w.r.t. DIRPE. RENE [16] also uses reflective Gray coding, adds additional bits to achieve no range expansion at all, but its applicability is limited to short ranges. Negation rules [40, 41] allow to define the opposite of a range via the opposite action (e.g., "deny" vs "accept"), so they are only applicable in specific scenarios.

VI. Conclusions

In this work, we introduce CGFE, a novel range encoding technique for TCAMs that combines the strengths of DIRPE and SRGE. CGFE achieves superior efficiency in encoding ranges of all sizes, significantly reducing the number of TCAM entries required compared to existing methods. This reduction directly translates to improved resource utilization and lower energy consumption, making CGFE an efficient solution for high-performance packet classification. Comprehensive analysis, including proofs of different CGFE properties and simulations, confirm the efficiency of CGFE across scenarios. In future research, we plan to explore enhancements to CGFE such as dynamic adjustment of chunk sizes, applications to more diverse network environments, and integration with other network optimization techniques to fully leverage its potential.

Acknowledgment

The authors would like to express their gratitude to the late Prof. Kirill Kogan for his invaluable advice and insightful discussions during the early stages of this work.

Appendix
Proof of Thm. 3

Proof: For a sequence of chunk sizes σ and range r we denote the CGFE encoding of r with chunk sizes σ as $\sigma(r)$. For sequences σ and σ' we say $\sigma \prec \sigma'$ holds iff $|\sigma(r)| \leq |\sigma'(r)|$ holds for all r. Thm. 3 is equivalent to for all σ, $\sigma \prec [1]^w$, where σ's total size is w and $[1]^w$ is a sequence of w chunks of size 1. The proof is by induction. Induction step takes the first chunk that has size greater than 1 and detaches one bit from the left, i.e., $[1]^i[c_1, c_2, \ldots, c_k] \prec [1]^{i+1}[c_1 - 1, c_2, \ldots, c_k]$. The induction step can be further reduced to $[c_1, c_2, \ldots, c_k] \prec [1, c_1 - 1, c_2, \ldots, c_k]$ as adding $[1]^i$ prefix can be shown to be monotonous w.r.t. $\prec$ due to chunk-by-chunk CGFE definition.

To prove the above, we assume an arbitrary range $r = [s, e]$ letting $\sigma^* = [c_1, \ldots, c_k]$, $\sigma = [v]\sigma^*$, $\sigma' = [1, v-1]\sigma^*$, $\sigma'' = [v-1]\sigma^*$, and $w = v + \sum_i c_i$. We need to show $|\sigma(r)| \leq |\sigma''(r)|$. We consider all possible cases for σ'. The case $\mathrm{MSC}_1(s) = \mathrm{MSC}_1(e)$ is straightforward since $\sigma'([\mathrm{TC}_1(s), \mathrm{TC}_1(e)])$ follows essentially the same steps as $\sigma([s, e])$. Henceforth, we assume $\mathrm{MSC}_1(s) = 0$ and $\mathrm{MSC}_1(e) = 1$. It is also straightforward if σ hits a middle range, i.e., $\mathrm{TC}_v(s) = 0$, $\mathrm{TC}_v(e) = 2^{w-v} - 1$ and $\sigma(r) = [\mathrm{MSC}_v(s), \mathrm{MSC}_v(e)] \circ \sigma^*([\ldots])$, where $[\ldots]$ is the range covering all possible values, hence $\sigma(r) = 1 \leq \sigma'(r)$. We look now at cases for $s'' = \mathrm{TC}_1(s)$ and $e'' = \mathrm{TC}_1(e)$.

Case $s'' = 0$ and $e'' = 2^{w-1} - 1$: σ's middle range.

Case $s'' = 0$ and $e'' < 2^{w-1} - 1$: as we already covered σ's middle range, we can assume $\mathrm{TC}_v(e) < 2^{w-v} - 1$. Both σ and σ' hit bottom range: $\sigma'(r) = \sigma'(r'_1) \cup \sigma'(r'_2)$ with $r'_1 = [s, x-1]$, $r'_2 = [x, e]$, $x = 2^{w-1}$ while $\sigma(r) = [0, MSC[v]e - 1] \circ \sigma^*([\ldots]) \cup \mathrm{MSC}_v(e) \diamond \sigma^-([0, \mathrm{TC}_v(e)])$. As the first part of $\sigma(r)$ has only one entry, we only need $|\sigma^-([0, \mathrm{TC}_v(e)])| \leq |\sigma'(r'_2)|$ for $\sigma'(r'_2) = 1 \diamond \sigma''(r''_2)$, $r''_2 = [0, e'']$. If $\mathrm{MSC}_{v-1}(e'') = 0$ then $\sigma''(r''_2) = 0 \diamond \sigma^*([0, \mathrm{TC}_v(e)])$, else $\sigma''(r''_2) = [0, \mathrm{MSC}_{v-1}(e'') - 1] \circ \sigma^*([\ldots]) \cup \mathrm{MSC}_{v-1}(e'') \diamond \sigma^*([0, \mathrm{TC}_v(e)])$ (bottom range).

Case $s'' > 0$ and $e'' = 2^{w-1} - 1$: symmetric.

Case $s'' > 0$ and $e'' < 2^{w-1} - 1$: σ' has a regular range with an odd $\mathrm{MSC}_1(e) - \mathrm{MSC}_1(s)$ and splits into $r'_1 = [s, 2^{w-1} - 1]$ and $r'_3 = [2^{w-1}, e]$. Assuming wlog. $|r'_1| \leq |r'_3|$, $\sigma'(r) = [0, 1] \circ \mathrm{TC}(0 \diamond \sigma''(r''_1)) \cup 1 \diamond \sigma''(\hat{r}''_2)$, where $r''_1 = [s'', \ldots]$ covers all values $\geq s''$ and $\hat{r}''_2 = [0 \ldots e''_1, e'']$ is *partial* for $r''_2 = [0, e'']$ with the non-covered start $e''_1 = 2^{w-1} - s''$. Now, we let $r^*_1 = [\mathrm{TC}_v(s), \ldots]$, $r^*_3 = [0, \mathrm{TC}_v(e)]$ and focus on $\sigma''(\hat{r}''_2)$ (§ III-D4).

Subcase $e''_1 > e''$: $\sigma''(\hat{r}''_2) = \emptyset$ $|r'_1| \leq |r'_3|$ implies $e''_1 = e'' + 1$, $|r'_1| = |r'_3|$, $|r^*_1| = |r^*_3|$, and odd $\mathrm{MSC}_v(e) - \mathrm{MSC}_v(s)$. $\mathrm{TC}_v(s) = 0$ and $|r^*_1| = |r^*_3|$ imply σ's middle range (covered), so let $\mathrm{TC}_v(s) > 0$. If $\mathrm{MSC}_{v-1}(s'') = 2^{v-1} - 1$ then $\sigma''(r''_1) = \mathrm{MSC}_{v-1}(s'') \diamond \sigma^*(r^*_1)$ and $\sigma(r) = [\mathrm{MSC}_v(s), \mathrm{MSC}_v(e)] \circ \mathrm{TC}(\mathrm{MSC}_v(s) \diamond \sigma^*(r^*_1))$ and we are done. If $\mathrm{MSC}_{v-1}(s'') < 2^{v-1} - 1$, $\sigma''(r''_1) = \mathrm{MSC}_{v-1}(s'') \diamond \sigma^*(r^*_1) \cup [\mathrm{MSC}_{v-1}(s'') + 1, \ldots] \circ \sigma^*([\ldots])$, while $\sigma(r) = [\mathrm{MSC}_v(s), \mathrm{MSC}_v(e)] \circ \mathrm{TC}(\mathrm{MSC}_v(s) \diamond \sigma^*(r^*_1)) \cup [\mathrm{MSC}_v(s) + 1, \mathrm{MSC}_v(e) - 1] \circ \sigma^*([\ldots])$, and we are also done.

Subcase $\mathrm{MSC}_{v-1}(e''_1) = \mathrm{MSC}_{v-1}(e'')$: $\sigma''(\hat{r}''_2) = \mathrm{MSC}_{v-1}(e'') \diamond \sigma^*([0 \ldots, \mathrm{TC}_{v-1}(e''_1), \mathrm{TC}_{v-1}(e'')])$ and here we have two further options: $\mathrm{TC}_{v-1}(e''_1) = 0$ and $\mathrm{TC}_{v-1}(e''_1) > 0$. In the former case, $\sigma^*([0 \ldots \mathrm{TC}_{v-1}(e''_1), \mathrm{TC}_{v-1}(e'')]) = \sigma^*(r^*_3)$, $\sigma''(r''_1) = [\mathrm{MSC}_{v-1}(s''), \ldots] \circ \sigma^*([\ldots])$, while $\sigma(r) = [\mathrm{MSC}_v(s), \mathrm{MSC}_v(e) - 1] \circ \sigma^*([\ldots]) \cup \mathrm{MSC}_v(e) \diamond \sigma^*(r^*_3)$, so we are done. In the latter case, $\sigma''(r''_1) = \mathrm{MSC}_{v-1}(s'') \diamond \sigma^*(r^*_1) \cup [\mathrm{MSC}_{v-1}(s'') + 1, \ldots] \circ \sigma^*([\ldots])$ *and* $|r^*_1| < |r^*_3|$, which implies that either $\sigma(r) = \mathrm{MSC}_v(s) \circ \sigma^*(r^*_1) \cup [\mathrm{MSC}_v(s) + 1, \mathrm{MSC}_v(e)] \circ \sigma^*([\ldots])$ (top range) or $\sigma(r) = \mathrm{MSC}_v(s) \diamond \sigma^*(r^*_1) \cup [\mathrm{MSC}_v(s) + 1, \mathrm{MSC}_v(e) - 1] \circ \mathrm{MSC}_v(e) \diamond \sigma^*([0 \ldots \mathrm{TC}_{v-1}(e''_1), \mathrm{TC}_{v-1}(e'')])$ (regular range); in either case we are done.

Subcase $\mathrm{MSC}_{v-1}(e''_1) < \mathrm{MSC}_{v-1}(e'')$: let us introduce $r'' = [e''_1, x'']$, $x'' = \mathrm{MSC}_{v-1}(e'') \cdot 2^{w-v} - 1$. If $|r^*_3| \geq |r''|$, $\sigma''(\hat{r}''_2) = [0, \mathrm{MSC}_{v-1}(e'')] \circ \mathrm{TC}(\mathrm{MSC}_{v-1}(e'') \diamond \sigma^*(r^*_3))$, otherwise $\sigma''(\hat{r}''_2) = \sigma''(r''_2)$. In the latter case, $\sigma'(r) = [0, 1] \circ \mathrm{TC}(0 \diamond \sigma''(r''_1)) \cup 1 \diamond \sigma''(r''_2)$, i.e., σ' encodes each part separately and it is easy to show that $|\sigma(r)| \leq |\sigma'(r)|$. In the remaining $|r^*_3| \geq |r''|$ case, we must have $\mathrm{MSC}_{v-1}(e''_1) =$

$\text{MSC}_{v-1}(e'')-1$, implying $\text{MSC}_v(e)-\text{MSC}_v(s)$ is even, $|r_1^*| < 2^{w-v}$ and $|r''| = 2^{w-v} - |r_1^*|$, and the latter implies $\sigma'(r_1') = \text{MSC}_{v-1}(s'') \diamond \sigma^*(r_1^*) \cup [\text{MSC}_{v-1}(s'') + 1, \ldots] \circ \sigma^*([\ldots])$. The above implies $|r_3^*| + |r_1^*| \geq 2^{w-v}$ which then implies $\sigma(r) = [\text{MSC}_v(s), \text{MSC}_v(e) - 1] \circ \text{TC}(\text{MSC}_v(s) \diamond \sigma^*(r_1^*)) \cup [\text{MSC}_v(s) + 1, \text{MSC}_v(e)] \circ \text{TC}(\text{MSC}_v(e) \diamond \sigma^*(r_3^*))$. ■

REFERENCES

[1] T. Fuchino, T. Harada, and K. Tanaka, "Accelerating packet classification via direct dependent rules," in *Proc. NoF*, 2021, pp. 1–8.

[2] C. Jung, S. Kim, R. Jang, D. Mohaisen, and D. Nyang, "A scalable and dynamic ACL system for in-network defense," in *Proc. SIGSAC*, 2022, pp. 1679–1693.

[3] S. Kim, C. Jung, R. Jang, D. Mohaisen, and D. Nyang, "A robust counting sketch for data plane intrusion detection," in *Proc. NDSS*, 2023.

[4] A. Hakiri, A. Gokhale, P. Berthou, D. C. Schmidt, and T. Gayraud, "Software-defined networking: Challenges and research opportunities for future internet," *Comput. Netw.*, vol. 75, pp. 453–471, 2014.

[5] M. Polverini, J. Galan-Jimenez, F. G. Lavacca, A. Cianfrani, and V. Eramo, "Dynamic in-network classification for service function chaining ready SDN networks," in *Proc. NoF*, 2019, pp. 74–81.

[6] M. Adiseshaiah and M. Sailaja, "A parallel decision-making design for highly speedy packet classification," *MICPRO*, vol. 99, p. 104 826, 2023.

[7] H.-T. Lin and P.-C. Wang, "TCAM-based packet classification for many-field rules of SDNs," *COMCOM*, vol. 203, pp. 89–98, 2023.

[8] D. E. Taylor, "Survey and taxonomy of packet classification techniques," *ACM Comput. Surv.*, vol. 37, no. 3, pp. 238–275, 2005.

[9] Y. Wan et al., "T-cache: Efficient policy-based forwarding using small TCAM," *IEEE/ACM Trans. Netw.*, vol. 29, no. 6, pp. 2693–2708, 2021.

[10] K. Kogan, S. Nikolenko, O. Rottenstreich, W. Culhane, and P. Eugster, "SAX-PAC (Scalable And eXpressive PAcket Classification)," in *Proc. SIGCOMM*, 2014, pp. 15–26.

[11] K. Lakshminarayanan, A. Rangarajan, and S. Venkatachary, "Algorithms for advanced packet classification with ternary CAMs," in *Proc. SIGCOMM*, 2005, pp. 193–204.

[12] V. Srinivasan, G. Varghese, S. Suri, and M. Waldvogel, "Fast and scalable layer four switching," in *Proc. SIGCOMM*, 1998, pp. 191–202.

[13] A. Bremler-Barr and D. Hendler, "Space-efficient TCAM-based classification using gray coding," in *Proc. INFOCOM*, 2007, pp. 1388–1396.

[14] H. Che, Z. Wang, K. Zheng, and B. Liu, "DRES: Dynamic range encoding scheme for TCAM coprocessors," *IEEE Trans. Comput.*, vol. 57, no. 7, pp. 902–915, 2008.

[15] A. Bremler-Barr, D. Hay, D. Hendler, and B. Farber, "Layered interval codes for tcam-based classification," in *Proc. SIGMETRICS*, 2008, pp. 445–446.

[16] A. Bremler-Barr, Y. Harchol, D. Hay, and Y. Hel-Or, "Encoding short ranges in TCAM without expansion: Efficient algorithm and applications," *IEEE/ACM Trans. Netw.*, vol. 26, no. 2, pp. 835–850, 2018.

[17] Z. Liao et al., "PT-tree: A cascading prefix tuple tree for packet classification in dynamic scenarios," *IEEE/ACM Trans. Netw.*, vol. 32, no. 1, pp. 506–519, 2024.

[18] J. Matousek, A. Lucansky, D. Janecek, J. Sabo, J. Korenek, and G. Antichi, "ClassBench-ng: Benchmarking packet classification algorithms in the OpenFlow era," *IEEE/ACM Trans. Netw.*, vol. 30, no. 5, pp. 1912–1925, 2022.

[19] K. Qiu, J. Yuan, J. Zhao, X. Wang, S. Secci, and X. Fu, "FastRule: Efficient flow entry updates for TCAM-based OpenFlow switches," *IEEE J. Select. Areas Commun.*, vol. 37, no. 3, pp. 484–498, 2019.

[20] J. Yang, J. Yang, K. Huang, H. Rong, and K. F. Li, "A compression approach to reducing power consumption of TCAMs in regular expression matching," *COMCOM*, vol. 70, pp. 86–94, 2015.

[21] K. Kannan and S. Banerjee, "Compact TCAM: Flow entry compaction in TCAM for power aware SDN," in *Proc. ICDCN*, 2013, pp. 439–444.

[22] C. Chen, D. Barrera, and A. Perrig, "Modeling data-plane power consumption of future internet architectures," in *Proc. CIC*, 2016, pp. 149–158.

[23] A. Ghiasian, "Impact of TCAM size on power efficiency in a network of OpenFlow switches," *IET Netw.*, vol. 9, no. 6, pp. 367–371, 2020.

[24] C. R. Meiners, A. X. Liu, E. Torng, and J. Patel, "Split: Optimizing space, power, and throughput for TCAM-based classification," in *Proc. ANCS*, 2011, pp. 200–210.

[25] M. Abbasi, S. Vakilian, A. Fanian, and M. R. Khosravi, "Ingredients to enhance the performance of two-stage tcam-based packet classifiers in internet of things: Greedy layering, bit auctioning and range encoding," *EURASIP J. Adv. Signal Process.*, vol. 2019, pp. 1–15, 2019.

[26] V. Demianiuk and K. Kogan, "How to deal with range-based packet classifiers," in *Proc. SOSR*, 2019, pp. 29–35.

[27] B. Agrawal and T. Sherwood, "Modeling TCAM power for next generation network devices," in *Proc. ISPASS*, 2006, pp. 120–129.

[28] H.-T. Lin, W.-H. Pan, and P.-C. Wang, "Packet classification using TCAM of narrow entries," *Technologies*, vol. 11, no. 5, p. 147, 2023.

[29] S. Murugesan and N. M. Sk, "A novel range matching architecture for packet classification without rule expansion," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 23, no. 1, pp. 1–15, 2018.

[30] S. Saleh, A. S. Goossens, T. Banerjee, and B. Koldehofe, "Towards energy efficient memristor-based TCAM for match-action processing," in *Proc. IGSC*, 2022, pp. 1–4.

[31] I. Ullah, Z. Ullah, and J.-A. Lee, "EE-TCAM: An energy-efficient SRAM-based TCAM on FPGA," *Electronics*, vol. 7, no. 9, p. 186, 2018.

[32] X. Yin, D. Reis, M. Niemier, and X. S. Hu, "Ferroelectric fet based tcam designs for energy efficient computing," in *Proc. ISVLSI*, 2019, pp. 437–442.

[33] T. Venkata Mahendra, S. Wasmir Hussain, S. Mishra, and A. Dandapat, "Energy-efficient precharge-free ternary content addressable memory (tcam) for high search rate applications," *IEEE Trans. Circuits Syst. I: Regul. Pap.*, vol. 67, no. 7, pp. 2345–2357, 2020.

[34] V. S. M. Srinivasavarma, S. Vidhyut, and N. M. S, "A TCAM-based caching architecture framework for packet classification," *ACM Trans. Embed. Comput. Syst.*, vol. 20, no. 1, pp. 1–19, 2021.

[35] C. R. Meiners, A. X. Liu, and E. Torng, "Bit weaving: A non-prefix approach to compressing packet classifiers in TCAMs," *IEEE/ACM Trans. Netw.*, vol. 20, no. 2, pp. 488–500, 2012.

[36] A. X. Liu, C. R. Meiners, and E. Torng, "TCAM razor: A systematic approach towards minimizing packet classifiers in TCAMs," *IEEE/ACM Trans. Netw.*, vol. 18, no. 2, pp. 490–500, 2010.

[37] C. R. Meiners, A. X. Liu, and E Torng, "Topological transformation approaches to TCAM-based packet classification," *IEEE/ACM Trans. Netw.*, vol. 19, no. 1, pp. 237–250, 2011.

[38] K. Kogan, S. I. Nikolenko, P. Eugster, A. Shalimov, and O. Rottenstreich, "Fib efficiency in distributed platforms," in *2016 IEEE 24th International Conference on Network Protocols (ICNP)*, 2016, pp. 1–10.

[39] Huan Liu, "Efficient mapping of range classifier into ternary-CAM," in *Proc HOTI*, 2002, pp. 95–100.

[40] O. Rottenstreich and I. Keslassy, "On the code length of TCAM coding schemes," in *Proc. ISIT*, 2010, pp. 1908–1912.

[41] O. Rottenstreich, R. Cohen, D. Raz, and I. Keslassy, "Exact worst case TCAM rule expansion," *IEEE Trans. Comput.*, vol. 62, no. 6, pp. 1127–1140, 2013.