Space-Efficient TCAM-based Classification Using Gray Coding *

Anat Bremler-Barr† Danny Hendler‡

September 19, 2010

Abstract

Ternary content-addressable memories (TCAMs) are increasingly used for high-speed packet classification. TCAMs compare packet headers against all rules in a classification database in parallel and thus provide high throughput unparalleled by software-based solutions. TCAMs are not well-suited, however, for representing rules that contain range fields. Such rules typically have to be represented (or *encoded*) by multiple TCAM entries. The resulting *range expansion* can dramatically reduce TCAM utilization.

A TCAM range-encoding algorithm $\mathcal{A}$ is *database-independent* if, for all ranges r, it encodes r independently of the database in which it appears; otherwise, we say that $\mathcal{A}$ is *database-dependent*. Typically, when storing a classification database in TCAM, a few dozens of so-called *extra bits* in each TCAM entry remain unused. These extra bits are used by some (both database-dependent and database-independent) prior algorithms to reduce range expansion.

The majority of real-life database ranges are short. We present a novel database-independent algorithm called *short range gray encoding* (SRGE) for the efficient representation of short range rules. SRGE encodes range endpoints as *binary reflected gray codes* and then represents the resulting range by a minimal set of ternary strings. To the best of our knowledge, SRGE is the first algorithm that achieves a reduction in range expansion in general, and a significant expansion reduction for short ranges in particular, without resorting to the use of extra bits.

The "traditional" database-independent technique for representing range entries in TCAM is *prefix expansion*. As we show, SRGE significantly reduces the expansion of short ranges in comparison with prefix expansion. We also prove that the SRGE algorithm's range expansion is at least as good as that of prefix expansion for *any* range.

Real-world classification databases contain a small number of unique long ranges, some of which appear in numerous rules. These long ranges cause high expansion which is not significantly reduced by any database-independent range encoding scheme that we are aware of, including SRGE. We introduce *hybrid SRGE*, a database-dependent encoding scheme that uses SRGE for reducing the expansion of short ranges and uses extra bits for reducing the expansion caused by long ones. Our comparative analysis establishes that *hybrid SRGE* utilizes TCAM more efficiently than previously published range-encoding algorithms.

This work also makes a more theoretic contribution. Prefix expansion for ranges defined by W-bit endpoints has worst-case expansion ratio of $2W - 2$. It follows from the work of Schieber et al. [1] that the SRGE algorithm has a slightly better worst-case expansion ratio of $2W - 4$. We prove that *any* independent TCAM encoding scheme has worst-case expansion ratio of at least W.

Keywords: TCAM, packet classification, range encoding, Gray code

[*] A preliminary version of this paper appeared in the proceedings of the *26th IEEE International Conference on Computer Communications (INFOCOM 2007)*, pp. 1388-1396.

[†] School of Computer Science, the Interdisciplinary Center.

[‡] Department of Computer Science, Ben-Gurion University.

1 Introduction

Packet classification is an indispensable building block of numerous Internet applications in the areas of routing, monitoring, security, and multimedia [2, 3, 4]. Network routers employ packet classification schemes to streams of incoming or outgoing packets in order to determine how each packet should be handled. The routers use a *classification database* that consists of a set of *rule*s (a.k.a. *filter*s). Each such rule specifies which actions to apply to each packet that *matches* the rule, actions such as whether the packet should be forwarded or dropped, whether it should be logged or not, etc.

In addition to specifying which actions to take, each rule also specifies a pattern that determines which packets will match it. These patterns are specified based on packet header fields, such as the source/destination addresses and the source/destination port numbers. A *key* is constructed from the corresponding fields of each packet header and is compared against the database rules. If the key matches the rule, then that rule is used to determine how the packet should be handled. Packet classification is often a performance bottleneck in the network infrastructure. It is therefore important to design packet classification solutions that scale to millions of key search operations per second.

Ternary content-addressable memory (TCAM) devices are increasingly used in the industry for performing high-speed packet classification. A TCAM is an associative memory hardware device which can be viewed as an array of fixed-width entries. Each TCAM entry consists of ternary digits, each of which can assume the values 0, 1, or $*$ ('don't-care'). When storing a classification database, each TCAM entry is associated with a single classification rule and specifies a pattern of packets that match the rule. Typically this leaves a few dozens *extra bits* in each entry which can be used by range-encoding schemes (see Sections 3.2, 3.3).

TCAMs enable parallel matching of keys, such as those derived from packet headers, against all entries. They can thus provide high throughput that is unparalleled by software-based solutions. When a key matches multiple TCAM entries, the TCAM returns the index of the first matching entry. This index is then used to locate the information specifying which actions to apply to the packet.

A single ternary digit in a TCAM device requires 10-12 transistors compared to only 4-6 required by a single SRAM bit [5]. This, together with the fact that TCAM market size is significantly smaller than SRAM market size, explains why TCAM devices are much more expensive than SRAMs and are a significant contributor to the cost of gigabit line cards. As the number of TCAM devices deployed worldwide is growing quickly, improving TCAM memory utilization has important practical implications.

A significant obstacle to the efficient use of TCAMs for packet classification is the fact that they are not well suited for representing rules that contain *range fields*, such as port fields. In general, such rules must be represented by multiple TCAM entries. The resulting *range expansion* can dramatically reduce the utilization of TCAM memory.

Traditionally, a range rule (that is, a rule that contains one or more range fields) is converted to an equivalent set of prefix rules (each of which can be directly stored to a single TCAM entry) by using a *prefix expansion* technique originally proposed by Srinivasan et al. [6]. Prefix expansion can be highly inefficient, however, and was shown to cause a 6-fold and more expansion of ranges that appear in real-life databases [2].

1.1 Our Approach and Contributions

This paper presents a novel algorithm for the efficient representation of short range rules in TCAM-based classification databases. Our work is motivated by the following observation, resulting from the analysis of a large real-life classification database consisting of more than $223K$ rules: *the large majority of the ranges used by classification rules are relatively short.* Specifically, the length of 60% of the unique ranges and 40% of all ranges is less than 20.

We present a novel algorithm called *short range gray encoding* (SRGE) for the efficient representation of short range rules. SRGE works by encoding numbers (both range endpoints and search keys) using *binary-reflected Gray code* (BRGC) and by then covering each resulting range with a set of ternary strings.

A *Gray code* is a binary encoding of a contiguous range of integers such that the codes of any two adjacent numbers differ by a single bit. An n-bit BRGC is constructed recursively by reflecting an $(n-1)$-bit BRGC (see Section 4 for more details). It is exactly this reflection property that is being used by the SRGE algorithm as it enables the construction of efficient range covers by using ternary strings that contain 'don't-care' symbols.

To illustrate the benefits of using BRGC codewords as opposed to a regular binary representation, consider a range $R = [i, i+1]$ of length 2. As the BRGC codes of i and $i+1$ differ by a single bit, R can be represented by a single TCAM entry that contains a single don't-care digit (in the single position where i and $i+1$ differ). In contrast, an average of 50% of length-2 ranges over numbers encoded by the regular binary representation require *two* TCAM entries.

Similarly to BRGC, previously published algorithms [7, 8, 9] also manage to reduce the expansion of rules (as compared to prefix expansion) by representing ranges as a set of arbitrary ternary values rather than as a set of prefixes. However, all these algorithms require extra bits, a small number of each is available in each TCAM entry, for representing *all ranges.* To the best of our knowledge, SRGE is the first algorithm that achieves a reduction in range expansion in general, and a significant expansion reduction for short ranges in particular, without resorting to the use of extra bits.

Another novel idea used by SRGE to further reduce expansion is the representation of ranges by a set of possibly *overlapping* ternary strings. All previously published algorithms use a set of non-overlapping strings to represent a range. We emphasize that, although some of the entries representing a range may overlap, *the SRGE algorithm only requires a single TCAM lookup.* If a key falls inside a range R, then the lookup will return the first matching entry that belongs to the cover of R.

As our empirical results show (see Section 6.3), SRGE achieves significant reduction in the number of redundant TCAM entries required to represent short ranges (when compared with prefix expansion) for most of the range rules. A similar reduction in expansion is obtained for randomly-generated short ranges.

Real-world classification databases contain a small number of unique long ranges, some of which appear in numerous rules. These long ranges cause high expansion which is not significantly reduced by any database-independent range encoding scheme that we are aware of, including SRGE. We introduce *hybrid SRGE*, a database-dependent encoding scheme that uses SRGE for reducing the expansion of short ranges and uses extra bits for reducing the expansion caused by long ones. Our comparative analysis establishes that *hybrid SRGE* utilizes TCAM more efficiently than previously published range-encoding algorithms. We emphasize that, unlike previously published algorithms, Hybrid-SRGE significantly reduces range expansion for a large majority of the ranges without having to use extra bits. We consequently believe that our algorithm will scale well into the future as the fraction of short ranges used by classification databases continues to increase, as is anticipated [9].

On the more theoretical side, we prove that if no extra bits are used, then the worst-case expansion of *any* range encoding scheme is at least W.

2 Terminology and Problem Statement

In this section we introduce the terminology we use throughout the paper and define the problem addressed by it. We follow the notation of [9] wherever appropriate.

A packet header consists of fields, each of which is a bit string. A *key* is a collection of K fields from the packet header. Keys are matched against classification *rules* stored in *entries* of a *classification database*.

Rules consist of K fields matching the corresponding key fields. Packet P matches rule R if each of P's K key fields matches the corresponding field of R. Each rule field f can specify one of three types of *matches*.

1. *Exact match*: field f matches key field g if they are equal.

2. *prefix match*: a *prefix* is a string of bits. Field f is a prefix match for key field g if g is a prefix of f.

3. *range match*: a *range* is a contiguous interval of integers $[s, e]$, where s and e are W-bit numbers and $s \leq e$. Key fields matched by ranges are port fields of constant size W (typically 16 bits). The *length* of a range is the number of integers it contains.

This paper deals with classification databases that reside in a TCAM device. A TCAM entry consists of ternary digits, each of which can assume the values 0, 1 or 'don't care', denoted by $*$. Each TCAM entry is wide enough to contain the concatenation of all the key fields, possibly leaving room for some *extra bits*.

If a rule consists solely of fields that specify exact and/or prefix matches then it can be represented by a TCAM entry in a straightforward manner: a field representing an exact match is stored in the TCAM entry as is; a field representing a prefix match is padded with the appropriate number of don't-cares in the least significant digits.

In general, rules containing one or more range fields cannot be represented by a single TCAM entry and *range encoding schemes* are used to encode each range as a set of TCAM entries. An encoding scheme maps each range R to a set of TCAM entries that represent it, called the *cover set of R*.

The *expansion* of a range is the number of TCAM entries in its cover set. The *range expansion factor* of an encoding scheme E, denoted R_E, is the maximum size of a cover set when using E, the maximum taken over all possible ranges. The expansion factor is a function of W, the number of bits required to represent range endpoints.

A widely used scheme for range encoding converts a range to a set of prefixes, each of which is stored at a separate TCAM entry. For example, for $W = 3$, the range $[1, 6]$ can be represented by the cover set $\{001, 01*, 10*, 110\}$ with range expansion 4. It is known that the range expansion factor of this scheme over W-bit ranges is $2W - 2$ [2].

Let D be a classification database. We let $n(D)$ denote the total number of rules in D. We let $n_E(D)$ denote the number of TCAM entries required to represent D using scheme E. Clearly $n_E(D) = n(D)$ if D contains no range rules. We let $F_E(D)$ denote D's *database expansion factor* using E, defined as $\frac{n_E(D)}{n(D)}$. In other words, $F_E(D)$ is the relative increase in the number of entries required to represent D in TCAM using scheme E.

We let $r(D)$ denote the number of range rules in D. We let $nr_E(D)$ denote the number of TCAM entries required to represent all of D's range rules using encoding scheme E. The *range redundancy* of an encoding scheme E for a range R is the number of additional, redundant, TCAM entries required to represent R when represented by E. We let $\mathit{FR}_E(D)$ denote D's *range redundancy factor* using E, defined as $\frac{nr_E(D)-r(D)}{r(D)}$. In other words, $\mathit{FR}_E(D)$ is the average number of redundant TCAM entries required to encode range rules of D using scheme E.

In this paper we propose an encoding scheme that reduces both the database expansion factor and the range redundancy factor for real-world classification databases.

3 Related Work

The issue of using TCAM devices for packet classification has received considerable attention from the research community over the last few years. A key question dealt with by researchers in this regard is that of improving the utilization of TCAM memory. This issue was considered both from the algorithmic [7, 8, 9, 10, 11] and the architectural [12] perspectives.

Spitznagel, Taylor, and Turner, introduced *Extended TCAM* (E-TCAM) [12], which implements range matching directly in hardware in addition to reducing power consumption by over 90% relative to standard TCAM. While this may represent a promising long-term solution, it seems that changing the ternary nature of TCAM entries while maintaining reasonable per-bit cost and addressing scalability issues will not be accomplished in the near future.

In this section we briefly describe prior algorithmic work that is related to the issue of TCAM range representation.

3.1 Prefix Expansion

The traditional technique for range representation, originated by Srinivasan et al. [6]), is to represent a range by a set of prefixes, each of which can be stored by a single TCAM entry. The worst-case expansion ratio when using prefix expansion for W-bit fields is $2W-2$. The problematic range is $R_w = [1, 2^w - 2]$. It is easily seen that the smallest set of prefixes that covers R_w is the following: $\{01*^{w-2}, 001*^{w-3}, 0001*^{w-4}, \cdots, 0^{w-1}1, 10*^{w-2}, 110*^{w-3}, \cdots, 1^{w-1}0\}$. [1]

As observed by [2], a single rule that includes two 16-bit range fields could, in the worst-case, require $(2 \cdot 16 - 2)^2 = 900$ entries.

3.2 Database-dependent Range Encoding

Database-dependent encoding of ranges makes use of extra bits, available in TCAM entries, for encoding ranges that occur in the database more efficiently. In such schemes, the encoding of a range may depend on the number of occurrences of that range (and of other ranges) in the database.

The first database-dependent algorithm is due to Liu [7]. The basic idea is to use the available extra bits as a bit map: a single extra bit is assigned to each selected range r in order to avoid the need to represent r by prefix expansion. If range r is assigned extra bit i, then the i'th extra bit is set in all TCAM entries that

[1] While this is the smallest set of prefixes required to cover R_w, we observe that this is *not* the smallest set of *ternary strings* that can cover R_w. In fact R_w can be covered by the following set of W ternary strings: $\{01*^{w-2}, *01*^{w-3}, **01*^{w-4}, \cdots, 1*^{w-2}0\}$.

include r; all other extra bits are set to 'don't care' in these entries. In the search key, extra bit i is set to 1 if the key falls in range r or set to 0 otherwise.

The number of unique range fields in today's classification databases is around 300. As observed by [9], this number is anticipated to continue to grow in the future. The number of extra bits per TCAM entry may vary according to the configuration of the device, but is typically a few dozen bits. It is therefore clear that the aforementioned simple scheme is not scalable.

Liu [7] proposed hierarchical encoding schemes to alleviate this problem. The encoding scheme of [7], however, may result in high expansion. Lunteren and Engbersen [8] suggested to use hierarchical encoding for compressing general TCAM rules. They present several versions of their scheme. However, the version that may reduce the expansion of range rules considerably complicates the task of incrementally updating the database.

In addition to making updates more expensive, hierarchical encoding requires extra logic so that the appropriate search key fields can be matched against all possible ranges. To maintain high throughput, either special-purpose hardware must be used or a pre-computed table must be stored in memory, whose size increases quickly with the number of ranges. These problems restrict the scalability of hierarchical dependent encoding.

Che et al. present a bit-map based scheme, called DRES, that employs a *dynamic range selection* algorithm for selecting the ranges that are to be assigned extra bits [13]. DRES is a greedy algorithm that assigns extra bits to the ranges with highest prefix expansion.

In contrast with these works, our hybrid-SRGE scheme encodes the vast majority of ranges without using extra bits and uses dependent encoding only for a very small number of ranges. It can therefore provide more scalability than these solutions.

Some past schemes, such as [14, 15, 16, 17, 18, 19, 20, 21, 22, 23], apply more general *classifier minimization* algorithms to the classification database rather than handle range expansion explicitly. These algorithms attempt to convert a classification database to an equivalent database that consumes fewer TCAM entries, using massive database preprocessing. Meiners et al. [24] present two novel classifier minimization approaches where classifier reencoding is viewed as a topological transformation. Their algorithms take rule decisions into account for improved compression and can use small TCAM tables to obtain faster reencoding. They report on between 3-fold and 8-fold space reduction as compared with previously published classifier minimization algorithms.

Zheng et al. [25] present DPPC-RE, a classification scheme that utilizes multiple TCAM devices. DPPC-RE extends a technique presented earlier by Zheng et al. [26] for dividing a classification database between multiple TCAM devices based on the values of carefully selected "ID bits". By appropriate selection of the ID bits, a classification database can be partitioned to several subsets of roughly the same size. Dynamic space- and load-balancing heuristics are then used to ensure high throughput and space efficiency. This technique is combined in [25] with the DRES algorithm [13] so that it can be applied across multiple TCAM devices.

3.3 Database-Independent Range Encoding

Database-independent encoding techniques are techniques that encode each range independently of the distribution of ranges in the database. Lakshminarayana et al. present a clever algorithm for the independent encoding of ranges, based on the concept of *fence encoding* [9]. Their technique, called Database-Independent Pre-Encoding (DIRPE), represents ranges by sets of arbitrary ternary strings and is based on the use of extra bits. Unlike the algorithms described in Section 3.2 , DIRPE is an independent encoding scheme and extra bits are never assigned to any particular range.

The efficiency of DIRPE is a function of the number of extra bits that are available. When the number of available extra bits decreases, the DIRPE-encoding of *all* ranges becomes less efficient. In contrast, the SRGE algorithm is an efficient database-independent encoding scheme that does not require extra bits.

Hybrid-SRGE encodes the few ranges whose expansion is not improved by SRGE by assigning an extra bit to each of them. A similar hybrid approach is employed by DIRPE for decreasing its expansion. However, unlike in our scheme, in the case of DIRPE the assignment of extra bits to frequently-occurring ranges increases the expansion ratio of all other ranges.

We evaluated our hybrid-SRGE scheme on the same database that was used to evaluate DIRPE in [9]. Our results establish that hybrid-SRGE outperforms DIRPE on that database by achieving a database expansion factor of 1.041, compared with a factor of 1.12 achieved by DIRPE. This superior expansion factor is achieved while using less than 40% of the extra bits used by DIRPE. Hybrid-SRGE's expansion factor is also significantly smaller than that of DRES.

3.4 Minimizing Boolean Expression

We observe that the problem of minimizing TCAM range expansion is, in fact, a special case of the problem of minimizing the size of Disjunctive Normal Form (DNF) expressions. This connection between the two problems was unnoticed prior to this work.

Mapping the TCAM range expansion problem to the problem of minimizing DNF expressions is done as follows. The variables in the DNF expression correspond to the W bits representing a range. The boolean DNF expression is the representation of the range. A range is expressed as a sum of *minterms*, each of which represents a number [2] and the goal is to find the minimal sum-of-products of the expression. For example, let us consider the range $R = [10, 11]$. Let b_0 denote the units digit and let b_1 denote the tenths digits. The sum of minterms corresponding to R is b_1b_0+b_1b_0' and the minimal sum of products is b_1 which corresponds to the ternary string $1*$.

DNF minimization is a well studied problem. The Karnaugh maps technique can be used to solve small instances of the problem involving up to 10 variables and the Quine-McCluskey algorithm can provide a general solution. As the problem is NP-complete, the Quine-McCluskey algorithm is impractical when the DNF formula involves a large number of variables.

A recent paper by Schieber et al. presents a linear time algorithm for finding the minimum size DNF expression corresponding to any range of binary-coded numbers [1]. They also show that the worst-case expansion of ranges over binary-coded numbers to arbitrary ternary strings is $2W - 4$, slightly better than the $2W - 2$ expansion factor achievable when only prefixes may be used.

The rest of the paper is organized as follows. Sections 4 and 5 describes the SRGE algorithm and hybrid scheme, respectively. In section 6, our empirical evaluation of SRGE establishes that it reduces the expansion of a large majority of the ranges as compared to prefix encoding. In Section 7, we prove the correctness of the SRGE algorithm. We also prove that it computes an optimal cover for BRGC-encoded ranges. In section 8, we prove that any encoding scheme has worst-case expansion ratio at least W. We conclude with a discussion in section 9.

[2] A minterm is also called a *standard product* or *canonical product term*. This is a term in which each variable appears exactly once.

```
**SRGE-cover**($[s_b, e_b]$) **returns** a set of ternary strings covering the range $[s_b, e_b]$

(1) $s \leftarrow$ BRGC encoding of $s_b$, $e \leftarrow$ BRGC encoding of $e_b$, $p \leftarrow$ least common ancestor of $s$ and $e$
(2) $pl \leftarrow$ rightmost leaf in $p$'s left subtree, $pr \leftarrow$ leftmost leaf in $p$'s right subtree.
(3) **if** $|[s, pl]| \leq |[pr, e]|$ # The other case is symmetric
    (3.1) $prefixes_1 \leftarrow$ prefix cover of $[s, pl]$
    (3.2) $i \leftarrow$ digit position corresponding to $p$'s left/right edges
    (3.3) $\forall q \in prefixes_1$: set $q$'s $i$'th digit to $*$. # $prefixes_1$ now also covers the mirror of $[s, pl]$ with regard to $p$
    (3.4) $s' \leftarrow pr + |[s, pl]|$ # $[s', e]$ is the sub-range of $[pr, e]$ not covered by $prefixes_1$
    (3.5) **if** $|[s', e]| = 0$ **return** $pefixes_1$
# We still need to cover $[s', e]$
(4) $p' \leftarrow$ least common ancestor of $s'$ and $e$, $pl' \leftarrow$ rightmost leaf in $p'$'s left subtree,
    $pr' \leftarrow$ leftmost leaf in $p'$'s right subtree
(5) **if** $|[pr', e]| \geq |[s', pl']|$: # Case I
    (5.1) $prefixes_2 \leftarrow$ prefix cover of $[pr', e]$
    (5.2) $i \leftarrow$ digit position corresponding to $p'$'s left/right edges
    (5.3) $\forall q \in prefixes_2$: set $q$'s $i$'th digit to $*$. # $prefixes_2$ now also covers the mirror of $[pr', e]$ with regard to $p'$
    (5.4) return $prefixes_1 \bigcup prefixes_2$
(6) **else** # CASE II: $|[s', pl']| > |[pr', e]|$
    (6.1) $prefixes_2 \leftarrow$ prefix cover of $[pr', e]$
    (6.2) $q \leftarrow$ the prefix corresponding to $p'$ left subtree
    (6.3) **return** $prefixes_1 \bigcup prefixes_2 \bigcup \{q\}$

**SRGE-construct-key**($b$) **returns** a search key for value $b$
**return** BRGC encoding of $b$
```

Figure 1: Pseudo-Code for the SRGE algorithm

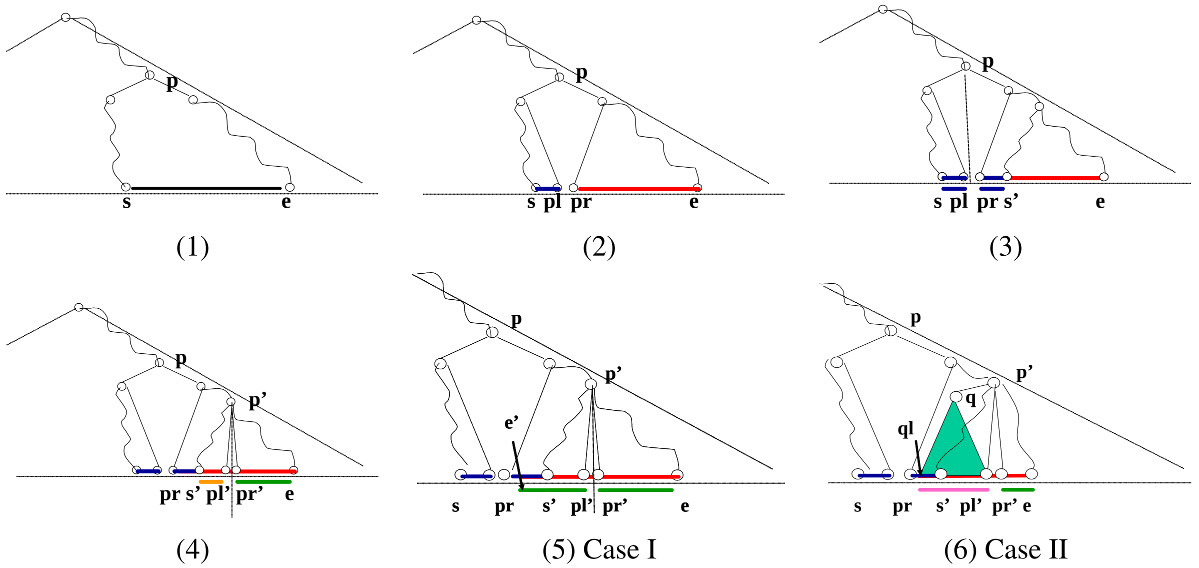
4 SRGE: Efficient Encoding of Short Ranges

In this section we describe the *Short Range Gray Encoding* (SRGE) algorithm for the efficient representation of short range rules in TCAM devices.

A *Gray code* is a binary encoding of a contiguous range of integers such that the codes of any two adjacent numbers differ by a single bit. SRGE uses a specific Gray code called the *binary-reflected Gray code* (BRGC). An n-bit BRGC is generated recursively as follows. The first 2^{n-1} code words are constructed by prefixing 0 to all the $(n-1)$-bit BRGC code words; the last 2^{n-1} code words are constructed by prefixing 1 to the reflected (i.e. listed in reverse order) list of $(n-1)$-bit BRGC code words. It is exactly this reflection property of the BRGC code that allows our algorithm to minimize the size of range cover sets.

Given an input range $[s_b, e_b]$ of numbers, where s_b and e_b respectively denote the binary-representations of the smallest and largest range numbers, the SRGE algorithm returns a *cover* $C_{[s_b,e_b]}$ of $[s_b, e_b]$ that consists of BRGC-encoded strings. The SRGE-encoding of any number within $[s_b, e_b]$ will be matched by some string of $C_{[s_b,e_b]}$. Moreover, no SRGE-encoding of a number outside $[s_b, e_b]$ will be matched by any of the strings of $C_{[s_b,e_b]}$. The concept of a cover is formally defined in the following.

Definition 1 *Let R be a range of integers. A strings-set C is a cover of R if any number in R is matched by at least one string of C and if no number outside of R is matched by a string of C. A prefix cover of R is a cover set of R that consists of prefixes. We let $CP(R)$ denote the prefix cover of R.*



(1) (2) (3)

(4) (5) Case I (6) Case II

Figure 2: The steps of the SRGE algorithm

The pseudo-code of the SGRE algorithm is shown in Figure 1. To facilitate pseudo-code understanding, we illustrate the operation of the algorithm in Figure 2, and provide an example of its operation on the range $[6, 14]$ in Figure 3. We refer to the algorithm steps depicted in Figure 2 in the detailed description of the algorithm which we now provide. We let $\mathcal{T}$ denote the full binary tree of height W shown in these illustrations. As shown in Figure 3, integer i (for $i \in [0, 2^w - 1]$) is represented by the i'th leaf of $\mathcal{T}$, which is labeled in Figure 3 by i'th BRGC encoding.

The SRGE-cover procedure receives a range $[s_b, e_b]$ of binary numbers, of size 2 or more. It returns a set of ternary strings covering the SRGE codes of all the numbers in this range.

First, the BRGC codes of s_b and e_b are computed (and stored into variables s and b, respectively); the least-common-ancestor (LCA) of s and b in $\mathcal{T}$, denoted p, is also computed (step (1), see Figure 1, (1)). Computing BRGC codes (not shown) is very simple and can be implemented efficiently in software as follows. The most significant digit is unchanged; any other digit i of the BRGC code is constructed by taking the exclusive-or of binary digits i and $i + 1$.

The range $[s, e]$ is split by p into two sub-ranges: one in p's left subtree and the other in its right subtree. The right and left endpoints of these sub-ranges are computed in step (2) and stored to variables pl and pr, respectively (see Figure 1, (2)).

Without loss of generality, assume that the sub-range in p's left subtree, $[s, pl]$, is no longer than the one in p's right subtree. A cover set of prefixes, denoted *prefixes*$_1$, that covers the BRGC codes of all the numbers in $[s, pl]$, is computed in sub-step (3.1). Now the reflection property of BRGC coding is used for minimizing the size of the cover set as follows (see Figure 1, (3)). The digit in each of the prefixes in *prefixes*$_1$ corresponding to p's right/left edges is changed to $*$ (sub-steps (3.2), (3.3)). The reflection property of BRGC coding guarantees that *prefixes*$_1$ now covers a mirror sub-range of $[s, pl]$ with regard to p.

If the input range $[s, e]$ is completely covered by *prefixes*$_1$, then this cover set is returned (sub-step (3.5)). Otherwise, the residue of the right sub-range still needs to be covered. Let the sub-range $[s', e]$ be that residue.

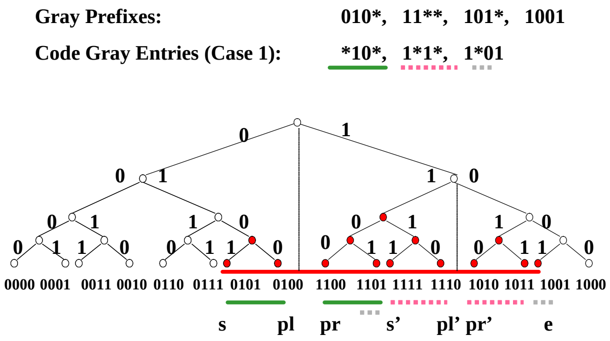


Figure 3: SRGE example: covering BRGC range $[6 - 14]$.

Similarly to step (3), the LCA of s' and e in $\mathcal{T}$, denoted p', is computed (step (4)). The range $[s', e]$ is partitioned by p' into two sub-ranges: one in its left subtree and the other in its right subtree. The left and right endpoints of these sub-ranges are computed and stored to variables pl' and pr', respectively (see Figure 1, (4)). Now there are two cases to consider.

1. The sub-range $[pr', e]$, in the right subtree of p', is no shorter than the sub-range $[s', pl']$ in the left subtree of p'. In this case we can cover $[s', pl']$ by reflecting the prefix cover set of $[pr', e]$ around p'. Note that we are guaranteed that all covered numbers are within the input range. This is accomplished by step (5), see Figure 1, (5).

2. Otherwise, reflecting the cover set of $[pr', e]$ is not enough to cover $[s', pl']$. In this case $[s', pl']$ is covered by the prefix corresponding to the left child of p'. Once again, we are guaranteed that all the numbers covered by this prefix are within the input range. This is accomplished by step (6), see Figure 1, (6).

We note that, in general, the ternary strings in the cover set produced by the SRGE-cover procedure may overlap each other. We emphasize that, although some of the entries representing a range may overlap, the SRGE algorithm only requires a single TCAM lookup. If a key falls inside a range R, then the lookup will return the first matching entry that belongs to the cover of R.

The SRGE-construct-key procedure receives a (binary) header field b and returns the corresponding key with which to search for rules matching the field (see Figure 1). The key is simply the SRGE encoding of b.

5 Hybrid-SRGE

The size of TCAM entries is typically larger than the size of the classification rules stored in them. This leaves a number of extra bits which can be used by range-encoding schemes. To further improve TCAM utilization, we use these extra bits by employing a database-dependent *hybrid-SRGE* scheme similar to that described in [7].

The high-level idea is to assign a single extra bit to each of the ranges whose TCAM entries consumption is highest under SRGE encoding. More specifically, let x denote the number of extra bits available in every TCAM entry. Hybrid-SRGE works as follows. First, it computes a list of the unique ranges that occur in the database, sorted in decreasing order of the overall number of redundant entries they require under SRGE encoding. By ‘overall number’ we mean the total number of redundant entries that are required by the SRGE representation of all the occurrences of the range in the database.

Then, each of the first x ranges in this list is dealt with by using a standard *database-dependent* encoding that assigns a single extra bit to it. We call these ranges the x *heaviest ranges*.

To exemplify hybrid-SRGE, consider the well-known range ≥ 1024 that encapsulates all dynamic/private ports. This is the heaviest range under SRGE. Hybrid-SRGE assigns the first extra bit (bit 1) to the heaviest range. Thus bit 1 is assigned to the range ≥ 1024. The extra bit is used as follows. Extra bit 1 of each entry that contains a source port field with the range ≥ 1024 is set to 1. Extra bit 1 of all other entries is set to $*$. [3]

As for search keys, bit 1 of a search key is set to 1 if the key falls within range R and to 0 otherwise. This guarantees that a key whose source port field is outside R will never match R and that a key whose source port field is inside R may match R, depending on the values of its other fields. Assigning an extra bit to a range results in expansion ratio 1 (i.e. no expansion) for that range.

Figure 4 presents an example that illustrates the operation of hybrid-SRGE. For simplicity, this example considers only a single packet field. The upper part of the figure shows four binary rules and their corresponding representation by BRGC strings. The bottom part of Figure 4 shows how these four rules are represented by hybrid-SRGE, when the first (least significant) extra bit is assigned to the range ≥ 011 and the second extra bit is assigned to the range ≥ 010. Observe that when an extra bit is assigned to a range, the extra bit is set to ’1’ while all other TCAM entry bits are set to ’*’. When the value ’010’ is being input, it is translated to the corresponding BRGC value - ’011’. Since this value falls within the range ≥ 011, the first extra bit of the search key is set to 1 and the search key will thus match the 3’rd entry.

6 Experimental Results and Comparative Analysis

We evaluated the efficiency of the hybrid-SRGE scheme on both real-life and random databases. Our real-life database is a collection of 126 separate files originating from various applications (e.g., firewalls, acl-routers, intrusion prevention systems) collected over the year 2004. This database is the union of the two databases that were used by [2] and [9]. The database comprises a total of 223K 144-bit rules which contain 280 unique ranges. Source-port fields contain 85 distinct ranges and destination-port fields contain 273 unique ranges (some ranges appear in both the source-port and destination-port fields).

Prefix expansion resulted in an expansion factor of 2.6. Overall, approximately 26% of the database rules contained range fields. Excluding the single range ≥ 1024, approximately 14% of the rules contained ranges.

6.1 Short Ranges

Figure 5 displays the distribution of range lengths in our database. More than 60% of the unique ranges that appear in the database have length less than 20 and 22% percentage of the total number of unique ranges have length 2. However, only 22% of the *total* number of ranges have length less than 20. The huge difference between the fractions of short unique ranges and short ranges is largely caused by a single rule that appears very frequently in a single database file.

[3] The range ≥ 1024 can appear also in the destination port field. All the occurrences of this range in the destination port field are represented independently by an additional extra bit, extra bit 2.

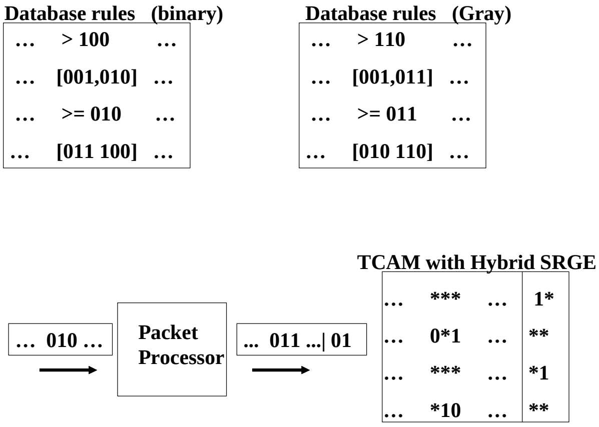


Figure 4: An example illustrating the operation of the hybrid-SRGE scheme.

To mitigate the effect of such anomalous files, Figure 5 displays also the fraction of short ranges when only ranges that appear in at least *two files* are taken into account. Measured this way, the fraction of short ranges grows to approximately 40%.

Why do short ranges occur so frequently in real-life classification databases? This phenomenon primarily results from the fact that ranges are commonly used for matching the source-port and destination-port fields. Port numbers are allocated by IANA [27] and it is often the case that ports that belong the same protocol family are assigned consecutive numbers. For example, the port number of $snmp$ is 161 and the port number of $snmptrap$ is 162.

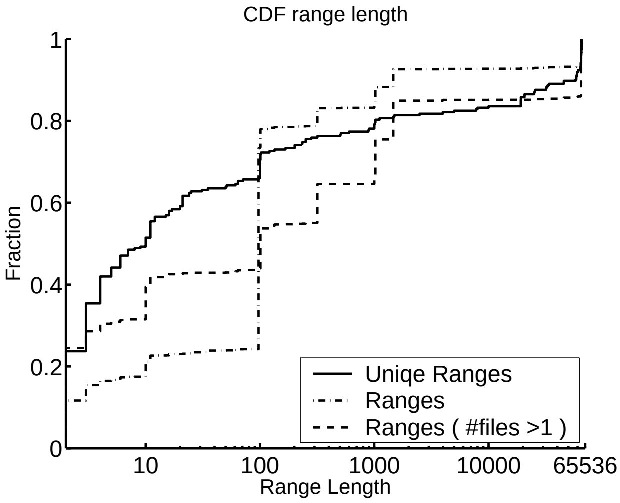


Figure 5: Distribution of range lengths, calculated as a fraction of 1) the total number of unique ranges, 2) the total number of ranges 3), the total number of ranges that appeared in at least 2 files

Algorithm	Extra bits	Redundancy	Expansion
Binary	2	3.37	1.69
Hybrid-binary (DRES)[13]	14	0.23	1.054
Hybrid-SRGE	14	0.15	1.041
Hybrid-DIRPE[9]	36	-	1.12

Table 1: Expansion and redundancy factors when using Binary (prefix expansion) and hybrid schemes on our real-life database. For all schemes, two extra bits are assigned for the range ≥ 1024 occurring in the source and destination port fields.

Many classification rules need to match the $snmp$ protocol family and, consequently, use the range $161 - 162$. Typically, each application is assigned a small number of ports which can thus be matched by a short range.

There is, however, also a small number of applications that use a wide range of ports. As two examples, Microsoft's DirectX gaming uses ports in the range $2300 - 2400$ and Real Audio uses ports in the range $6970 - 7170$ [4].

A second type of long ranges are ranges that partition all ports to two general categories [28]. Key examples are the well known rules < 1024 and ≥ 1024 that partition ports to the sets of registered ports and dynamic/private ports, respectively.

6.2 Evaluating SRGE on Random Databases

We compare the efficiency of different range representation algorithms by comparing their *database expansion factor* and *range redundancy factor*. The database expansion factor of database *D* using scheme *E* is the relative increase in the number of entries required to represent D in TCAM using scheme E. The *range redundancy factor* of database *D* using scheme *E* is the average number of redundant TCAM entries required to encode range rules of D using E. See Section 2 for the precise definitions of these metrics.

Clearly, a perfect encoding scheme will achieve a database expansion factor of 1 (i.e. no expansion). The range encoding redundancy factor focuses only on range rules and quantifies the number of extra TCAM entries that is required to represent them. A perfect encoding scheme will clearly have a range encoding redundancy factor of 0.

Figure 6 compares the efficiency of SRGE and prefix encoding on a database that consist of ranges whose length is chosen randomly and uniformly from the interval $[2, 2^x]$.[5] It shows the reduction in range expansion obtained by SRGE as compared with prefix expansion, as a function of x. As shown by Figure 6, SRGE is significantly more efficient for short ranges. For example, for ranges of length 2, SRGE reduces the expansion by 33% on average; for ranges of length 64 ($x = 6$) or less, the average reduction is slightly more than 20%.

6.3 Evaluating hybrid-SRGE on Real-Life Databases

Table 1 summarizes the results of the evaluation we conducted on our real-life database. This is the same database used by DIRPE [9]. We evaluated four schemes: Binary (prefix expansion), Hybrid-binary as

[4] For confidentiality reasons, the above examples are not derived from our database.

[5] The minimum length of a range is 2.

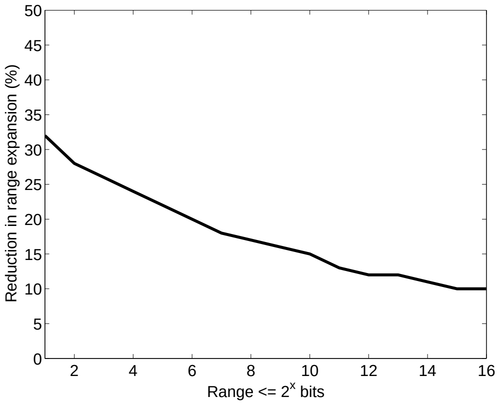


Figure 6: Random database: reduction in range expansion achieved by SRGE as a function of range length.

implemented by DRES [13], Hybrid-DIRPE and Hybrid-SRGE. For all schemes, we assign two extra bits to represent the occurrences of the well-known ≥ 1024 rule in the source and destination fields. It can be seen that using Binary results in a high redundancy factor and an unacceptable expansion factor of 1.69. As indicated by the right-hand side of Figure 8, SRGE using only two extra bits does not significantly improve over Binary using two extra bits.

When 12 extra bits are used in addition to the two bits representing the ≥ 1024 ranges, DRES reduces redundancy to 0.23, approximately 50% more than Hybrid-SRGE whose redundancy is only 0.15 using the same number of bits. Database size expands by 5.4% with DRES and by 4.1% with Hybrid-SRGE. Based on the data provided in [9], applying the Hybrid-DIRPE scheme results in a significantly larger expansion factor (1.12), even when it uses a much larger number of extra bits (36).

File Name	Lines Number	# dest ranges	#source ranges	Expansion DRES	Expansion Hybrid-SRGE	Redundancy DRES	Redundancy Hybrid-SRGE	reduction redundancy
acl1	9999	35	1	1.060606	1.043804	0.149852	0.108309	27.7%
acl2	9904	3	1	1.0	1.0	0	0	N/A
acl3	9995	38	1	1.024112	1.005703	0.066759	0.015789	76.3%
acl4	9992	50	1	1.035328	1.015312	0.106133	0.046001	56.7%
acl5	10000	5	1	1.0	1.0	0	0	N/A
fw1	9749	3	3	1.0	1.0	0	0	N/A
fw2	9871	1	2	1.0	1.0	0	0	N/A
fw3	9592	3	3	1.0	1.0	0	0	N/A
fw4	9576	6	6	1.0	1.0	0	0	N/A
fw5	9552	4	3	1.0	1.0	0	0	N/A
ipc1	9987	9	7	1.021428	1.0	0.033349	0	100%
ipc2	10000	1	1	1.0	1.0	0	0	N/A

Table 2: Expansion and redundancy factors when applying the Hybrid-Binary (DRES) and Hybrid-SRGE algorithms, using 14 extra bits, on ClassBench files.

Taylor's ClassBench [29] - a suite of tools for benchmarking packet classification algorithms - includes 12 real-life classification files provided by Internet Service Providers (ISPs). These files are of 3 types:

Algorithm	Extra bits	Expansion
Hybrid-binary (DRES)[13]	8	1.12
DIRPE[9]	36	1.12
Hybrid-SRGE	7	1.12
Hybrid-binary (DRES)[13]	219	1
Hybrid-SRGE	191	1

Table 3: The number of extra bits required by each of Hybrid-binary, Hbyrid-SRGE and DIRPE for obtaining the same expansion factor (1.12).

access control lists (files acl1-acl5), firewalls (fw1-fw5), and IP Chains (ipc1 and ipc2). Table 2 presents and compares the results of applying DRES and Hybrid-SRGE to these files with 14 extra bits. Observe that both DRES and Hybrid-SRGE incur no expansion on 8 of these files in which the number of ranges is small. On files with a larger number of ranges, Hybrid-SRGE reduces redundancy by a (per-file) average of 65% compared with DRES. [6]

Yet another way to compare range-encoding schemes that use extra bits is to fix an expansion factor and check the minimum number of extra bits that are required by each scheme to obtain this factor. This data is shown in the first 3 lines of Table 3 for the expansion factor reported for DIRPE [9] when it uses 36 bits. Both DRES and Hybrid-SRGE require significantly less extra bits than DIRPE for obtaining the same expansion factor; Hybrid-DIRPE requires one extra bit less than DRES. Lines 4-5 of Table 3 show the number of extra bits required by DRES and Hybrid-SRGE for reaching expansion factor 1.[7] Hybrid-SRGE requires 28 less extra bits.

Figure 7 displays the contribution of the first x ”heaviest ranges” (excluding the range ≥ 1024) to the total expansion. For each range, we calculate its contribution as the product of the number of times it appears in the database and the number of entries required to represent it by prefix expansion. As can be seen, less than 10 of the heaviest ranges contribute approximately 92% of the expansion.

Taking into account only ranges that appear in more than one file does not change the result significantly. The expansion factor of 1.07 was obtained by hybrid-SRGE by coding the 12 heaviest ranges with extra bits. [8] Our calculations show that SRGE reduces the redundancy caused by all the ranges that are not assigned extra bits by approximately 25%.

[6] Files were generated by running the db_generator utility on ClassBench seed files using the following parameters: *filters*=10000, *smoothness*=2, *address-scope*=-0.5, *port-scope*=0.1.

[7] We could not obtain this data for DIRPE.

[8] Two of the 10 unique ranges appear in both the source and destination port fields. We thus needed a total of 12 extra bits to encode these 10 ranges. Two additional bits were required for encoding the range ≥ 1024.

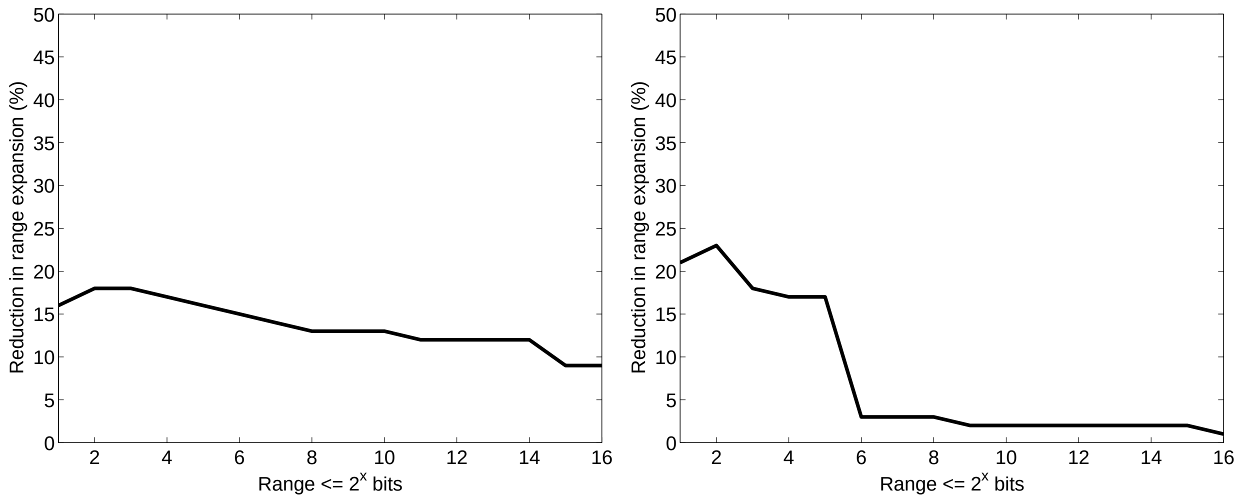


Figure 8: Real-life database: reduction in range expansion achieved by SRGE as a function of range length. Left-hand graph shows reduction when each range has weight 1. Right-hand graph shows reduction when range weight equals the number of its occurrences in the database.

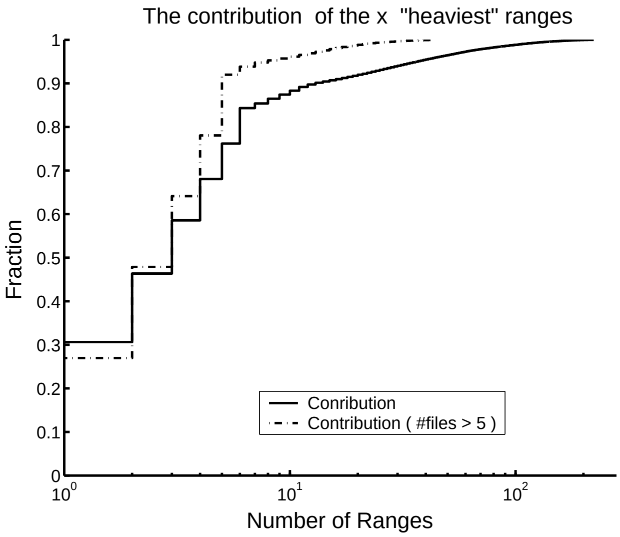


Figure 7: Distribution of the contribution of the x heaviest ranges to the overall expansion

Figure 8 is the equivalent of Figure 6. It shows the reduction in range expansion obtained by SRGE on ranges that appear in our real-life database, as a function of their length. The left-hand side shows the reduction when all ranges have the same weight. SRGE reduces expansion by more than 9% for all range lengths and by more than 15% for ranges of length 16 or less. SRGE's improvement declines moderately with the range length.

The right-hand side of Figure 8 shows the reduction when the number of occurrences of each range in the database is taken into account. As can be seen, SRGE fairs well on frequently-occurring short ranges of lengths 32 or less and reduces their average expansion by between 17% and 24%. However, there are several frequently-occurring long ranges for which SRGE's expansion reduction is relatively small, as can be seen by the sharp decline of the curve for ranges whose length exceeds 32.

7 SRGE correctness and complexity

In this section we prove that the SRGE algorithm correctly computes a minimum-size cover of its input range. We also analyze its time complexity.

We start by proving that the SRGE algorithm is correct and optimal, that is, it computes a minimum-size cover of its input range. Let T_W denote the full binary tree of height W, where leaves represent the numbers $0 \ldots 2^W - 1$. In the following definitions and proofs, whenever we mention a subtree, we refer to a subtree of T_W.

We use the following notations. Let R be an interval. We denote by $\mathcal{T}(R)$ the smallest subtree that contains R. For a tree T, we let *LST(T)* and *RST(T)* denote the left and right sub-trees of T, respectively.

When numbers are binary-encoded, each subtree is uniquely represented by a prefix. In other words, there is a single prefix p, such that all the leaves of the subtree are matched by p. The following claim states that the same is true also when BRGC-encoding is used.

Claim 1 *When numbers are BRGC-encoded, any subtree $\mathcal{T}$ of T_W matches some unique prefix.*

Proof: We prove the claim by induction. The claim clearly holds for $W = 1$. Assume the claim holds for $W = k$ and prove for $k + 1$. Recall that the BRGC code is constructed recursively: the first 2^k number-codes in T_{k+1} are constructed from the number-codes of T_k be prepending 0 to each of them. The last 2^k number-codes are constructed by reversing the order of the number-codes of T_k and prepending 1 to each of them.

Let $\mathcal{T}$ be a subtree of T_{k+1}. From induction hypothesis, there is some prefix p that matches all the leaves of $\mathcal{T}$ in T_k. From the recursive BRGC-code construction, if $\mathcal{T}$ is in the left subtree of T_{k+1}, then the prefix $0p$ matches $\mathcal{T}$ in T_{k+1}. Also, If $\mathcal{T}$ is in the right subtree of T_{k+1}, then the prefix $1p$ matches $\mathcal{T}$ in T_{k+1}. Finally, the left (respectively, right) subtree of T_{k+1} is matched by the prefix '0' (respectively, by the prefix '1').

Claim 1 is used by both the SRGE algorithm and the following correctness proofs, as it shows that, regardless of whether binary-encoding or BRGC-encoding is employed, every subtree can be covered by a single prefix. The following definitions are required for our proofs.

Definition 2 *A* prefix cover of R *is a cover set of R that consists of prefixes. We let $CP(R)$ denote the prefix cover of R.*

Definition 3 *Let R and B be two ranges and let C be a strings set. We say that* C covers R with background B, *if any number in R is matched by at least one string of C and if no number outside $R \cup B$ is matched by a string of C. We note that this generalizes the definition of a "regular" cover, since such a cover can be viewed as having background ϕ.*

Definition 4 *Let x be a ternary string and let $\mathcal{T}$ be a subtree. The* projection of x on $\mathcal{T}$ *is the set containing the leaves of $\mathcal{T}$ that are matched by x.*

Let p the root of $\mathcal{T}$. It is easily seen that the projection of x on $\mathcal{T}$ can be obtained from x by setting any '*'-digit of x that corresponds to a parent of p (if there are such digits) to the proper digit (0 or 1) corresponding to p.

Definition 5 *We say that a range R is* extremal *if R contains either the leftmost and/or the rightmost leaf of $\mathcal{T}(R)$.*

Lemma 1 *If R is an extremal range, then the prefix cover of R is optimal.*

Proof: We prove the claim by induction on the height of $\mathcal{T}(R)$. The claim holds trivially for the base case, where $\mathcal{T}(R)$ and R consist of a single number. Assume the claim holds for height k and prove for $k+1$. Let $R = [s, e]$ and let l and r respectively denote the leftmost and rightmost leaves of $\mathcal{T}(R)$.

If $s = l \wedge e = r$ holds, then the claim follows trivially, since, in this case, the prefix cover of R consists of the single prefix corresponding to $\mathcal{T}(R)$. Otherwise, let $[s, pl]$ and $[pr, e]$ denote the sub-ranges of R contained in $LST(\mathcal{T}(R))$ and $RST(\mathcal{T}(R))$, respectively. From the definition of $\mathcal{T}(R)$, $(pl \geq s) \wedge (pr \geq e)$ holds. We consider the following cases.

- Assume $pl - s = e - pr$ holds. Let p denote the root node of $\mathcal{T}(R)$. Let C' denote the prefixes-set obtained from $CP([s, pl])$ by replacing the digit corresponding to p with '*' in each of the prefixes of $CP([s, pl])$. From the reflection property of the BRGC code, C' is a cover of both $[s, pl]$ and $[pr, e]$, thus it is a cover of R. From induction hypothesis, $CP([s, pl])$ is an optimal cover of $[s, pl]$, hence C' is an optimal cover of R.

- Otherwise, $pl - s \neq e - pr$ holds. Without loss of generally, assume that $pl - s < e - pr$ holds. Since R is extremal, the sub-range $[pr, e]$ consists of all the leaves of $RST(\mathcal{T}(R))$. Let C' denote the prefixes-set obtained from $CP([s, pl])$ by adding the prefix corresponding to $RST\big(\mathcal{T}(R)\big)$. Clearly, C' is a prefix cover of R of size $|CP([s, pl])| + 1$.

  We claim that C' is an optimal cover of R. Assume otherwise, then there exists a cover C'' of R such that $|C''| \leq |CP([s, pl])|$. From the properties of the BRGC code and the '*' symbol, each string of C'' either does not cover any number in $[s, pl]$ or covers an equal number of leaves in $[s, pl]$ and in $[pr, e]$. Thus, there is at least one string $x \in C''$ that does not cover any leaves of $[s, pl]$. We can thus obtain from C'' a cover C^* of $[s, pl]$, by removing x and replacing all other strings with their projection on $LST\big(\mathcal{T}(R)\big)$. Clearly, $|C^*| < |CP([s, pl])|$. This contradicts the induction hypothesis, however, since $[s, pl]$ is extremal.

In the following we consider a range $R = [s, e]$ provided as input to the SRGE algorithm. We let p denote the root of $\mathcal{T}(R)$. We let pl and pr respectively denote the rightmost leaf of the left subtree of $\mathcal{T}(R)$ and the leftmost leaf of the right subtree of $\mathcal{T}(R)$. Without loss of generality, we assume $|[s, pl]| \leq |[pr, e]|$

Lemma 2 *Let s^* be the reflection of s w.r.t. p. Also, let D be a cover of R and let $D' \subseteq D$ be the set consisting of all strings in D that match some number in $[s, pl]$. Then the following hold.*

1. $|D'| \geq |CP([s, pl])|$.

2. $D' \cap [pr, e] \subseteq [pr, s^*]$.

Proof: The first claim follows from Lemma 1. As for the second claim, let $x \in D'$ be a string that matches leaf $pr + \delta$, for some $\delta \geq 0$. From the properties of the BRGC code and the '*' symbol, x also matches leaf $pl - \delta$. It follows that $\delta \leq pl - s$ must hold, hence $pr + \delta \leq s^*$ also holds.

The following corollary follows immediately from Lemma 2.

Corollary 1 *After performing lines (3.1)-(3.5) of the SRGE algorithm, variable* prefixes_1 *stores a strings-set that covers both* $[s, pl]$ *and* $[pr, s^*]$*. Moreover, there does not exist a cover of* $[s, pl]$ *with background R, with size* $|\text{prefixes}_1|$ *or smaller, that covers additional leaves.*

If $s^* = e$, then, clearly, the algorithm returns an optimal cover. Assume otherwise and let $s' = s^* + 1$. To complete the optimality proof, we must prove that the rest of the BRGC algorithm produces an optimal cover of $[s', e]$.

Lemma 3 *The strings-set computed in lines 4-6 of the BRGC algorithm is a minimum-size cover of* $[s', e]$ *with background R.*

Proof: Let p' denote the root of $\mathcal{T}([s', e])$, and let pl' (respectively, pr') denote the rightmost leaf of the left subtree of $\mathcal{T}([s', e])$ (respectively, the rightmost leaf of the left subtree of $\mathcal{T}([s', e])$). We consider the two cases handled by the algorithm.

1. $|[pr', e]| \geq |[s', pl']|$: in this case, the algorithm computes in line 5.1 the prefix cover of $[pr', e]$ and stores it to variable *prefixes*$_2$. Then, in lines 5.2 and 5.3, each of the strings in *prefixes*$_2$ is modified so that it also covers the mirror w.r.t p'.

   Clearly, after line 5.3 is performed, the set *prefixes*$_2$ is a cover of $[s', e]$ with background R. Assume in contradiction that that there exists a strings-set C that covers $[s', e]$ with background R, such that $|C| < |$*prefixes*$_2|$ holds. Let C' be the strings-set obtained from C by replacing each $x \in C$ with the projection of x on $\mathcal{T}([s', e])$. Since no string of C is allowed to match leaves to the right of e, C' is a cover of $[s', e]$ and $|C'| < |prefixes_2|$. This contradicts Lemma 1.

2. $|[pr', e]| < |[s', pl']|$: in this case, in lines 6.1 and 6.2, *prefixes*$_2$ is set to the union of the prefix cover of $[pr', e]$ and the prefix corresponding to the left sub-tree of p'. Clearly, *prefixes*$_2$ is a cover of $[s', e]$ with background R. Optimality follows from Lemma 1 and from the fact that nodes to the right of e must not be matched.

Theorem 1 *The SRGE algorithm computes a correct cover of minimum size.*

Proof: Follows from Corollary 1 and Lemma 3.

Theorem 1 establishes that the SRGE algorithm computes a correct and optimal cover C_R for the BRGC representation of its input range R. It is easy to show that any prefix cover of the *binary representation* of R is at least as large as C_R.

Lemma 4 *Let R be a range, let* C_R *be the cover returned by SRGE algorithm and let* C'_R *be the prefix cover of the binary representation of R, then* $|C_R| \leq |C'_R|$ *holds.*

Proof: From claim 1, every subtree is matched by a unique prefix, regardless of whether or not numbers are binary-encoded or BRGC-encoded. Hence, C'_R must include a prefix per every subtree of $\mathcal{T}(\mathcal{R})$ covered by C_R, whereas C_R may coalesce some of these prefixes by using the BRGC-code reflection property.

Lemma 4 establishes that, for any input ranges R, the cover computed by the SRGE algorithm is no larger than R's prefix cover. The next lemma analyzes the time complexity of the SRGE algorithm.

Lemma 5 *The time complexity of the SRGE algorithm is $O(W)$.*

Proof: Translating a number's encoding from binary to SRGE takes $O(W)$ operations, as does finding the least-common-ancestor of two leaves in T_W. In addition to these operations, the SRGE algorithm also computes the prefix covers of two intervals (in lines 3.1, and 5.1 or 6.1); each of these steps also requires $O(W)$ operations. Finally, from Lemma 4, the number of operations performed on the prefixes of the computed prefix covers (in lines 3.3. and 5.3) is also $O(W)$.

8 A Lower bound on Range Expansion Without Extra Bits

In this section we prove that any range encoding scheme that does not use extra bits has worst-case expansion ratio of at least W, regardless of the number encoding it uses.

A *W-encoding* is a 1:1 mapping of the integers in $\{0, \cdots, 2^W - 1\}$ onto the set of binary strings of length W. A string *matches* a set of ternary strings $\mathcal{P}$ if it matches at least one string of $\mathcal{P}$.

Lemma 6 *Let $\mathcal{P}$ be a set of ternary strings. If exactly $2^W - 1$ W-bit numbers match $\mathcal{P}$ then $|\mathcal{P}| \geq \mathtt{W}$ holds.*

Proof: The proof goes by induction. For $W = 1$, clearly, a set of ternary strings that is matched by either 0 or 1 must be of length at least 1. Assume the claim holds for $W = i$, we now prove for *W=i+1*. Let $\mathcal{P}$ be a set of ternary strings that is matched by exactly 2^{i+1}-1 *(i+1)*-bit binary numbers. Assume that $|\mathcal{P}| < \mathtt{i+1}$ holds to obtain a contradiction. Let $b_i b_{i-1} \cdots b_0$ be the single *(i+1)*-bit number that does not match $\mathcal{P}$. Consider the bits in position *i* of the strings in $\mathcal{P}$. If all these bits are in $\{b_i, *\}$, then either $(1 - b_i)b_{i-1} \cdots b_0$ does not match $\mathcal{P}$ or $b_i b_{i-1} \cdots b_0$ matches $\mathcal{P}$. Both these cases contradict our assumptions. Thus there must be at least one string in $\mathcal{P}$ whose most significant bit is $1 - b_i$. Let $\mathcal{P}'$ be the set obtained from $\mathcal{P}$ by removing all such strings from $\mathcal{P}$ and truncating the most significant bit of all remaining strings. Then $\mathcal{P}'$ covers all i-bit numbers except for $b_{i-1} \cdots b_0$ and is of length less than i. This is a contradiction.

Lemma 7 *The worst-case expansion ratio of any scheme is at least W, regardless of the encoding.*

Proof: From Lemma 6, the range $[0, \cdots, 2^W - 2]$ cannot be represented by a strings set of size less than W.

9 Discussion

We have presented the SRGE algorithm for the efficient encoding of short ranges. SRGE uses binary-reflected Gray code (BRGC) number encoding instead of binary number encoding and exploits the properties of BRGC encoding for reducing expansion. The SRGE algorithm achieves significant reduction in short-range expansion without resorting to the use of extra bits. To the best of our knowledge, SRGE is the first database-independent TCAM range-encoding scheme that is able to reduce expansion and redundancy without relying on extra bits. As we prove, SRGE computes an optimal cover for all BRGC-encoded ranges.

We've also introduced the Hybrid-SRGE scheme, which represents a small number of large high-expansion ranges by extra bits while using SRGE to reduce the expansion of the remaining ranges. Hybrid-SRGE significantly reduces the range expansion of a large real-life database in comparison with similar schemes such as DRES [13] and DIRPE [9]. We emphasize that we are not aware of any database-independent range encoding scheme that can significantly reduce the expansion of long-ranges without using extra bits.

Hybrid database-independent range-encoding schemes, that represent a few long ranges by using extra bits, support faster incremental classifier updates than database-dependent schemes whose encoding depends fundamentally on classifier contents. Hybrid-SRGE is more scalable than prior art hybrid database-independent schemes. This is because small ranges, which constitute the majority of the ranges occurring in today's real-life classification databases, can be efficiently encoded by SRGE without using extra bits. SRGE packet processing time is small, since the translation of a W-bit binary-encoded number to a W-bit BRGC-encoded number requires only W exclusive-or operations.

We have shown a lower bound of W on the worst-case expansion ratio of any ternary encoding scheme. It is known that the worst-case expansion ratio of prefix expansion is $2W-2$. By using a proof technique similar to that of [1], it can be shown that the worst-case expansion ratio of BRGC-encoded ranges is $2W-4$. We conjecture that there do exist ternary encoding schemes with better worst-case expansion ratio. Finding the tight bound on the worst-case expansion ratio of ternary encoding schemes remains an interesting open problem.

10 Acknowledgments

The authors are indebted to Cisco Systems, Will Eatherton, and David Taylor for kindly providing us the classification database we used for this work. We would also like to thank Yehuda Afek, Karthik Lakshminarayanan, Anand Rangarajan, and Srinivasan Venkatachary for assisting us in the process of obtaining access to the database. We are also indebted to Ronny Roth for pointing out the connection between the TCAM range encoding and DNF expression minimization problems and for helpful discussions. Finally, we would like to thank the anonymous reviewers for providing many helpful comments on an earlier draft of this paper.

References

[1] Baruch Schieber, Danny Geist, and Ayal Zaks, "Computing the minimum dnf representation of boolean functions defined by intervals," *Discrete Applied Mathematics*, , no. 1-3, pp. 154–173, 2005.

[2] D.E. Taylor, "Survey and taxonomy of packet classification techniques," *ACM Computer Surverys*, pp. 238–275, 2005.

[3] Pankaj Gupta and Nick McKeown, "Algorithms for packet classification," in *IEEE Network Special Issue*, 2001.

[4] George Varghese, *Network Algorithmics: An Interdisciplinary Approach to Designing Fast Networked Devices*, The Morgan Kaufmann Series in Networking, 2005.

[5] Florin Baboescu, Sumeet Singh, and George Varghese, "Packet classification for core routers: Is there an alternative to cams," in *INFOCOM*, 2003.

[6] V. Srinivasan, G. Varghese, S. Suri, and M. Waldvogel, "Fast and scalable layer four switching," in *ACM SIGCOMM 98*, Sept. 1998, pp. 191–202.

[7] H. Liu, "Efficient mapping of range classifier into ternary-cam," in *Hot Interconnects*, 2002.

[8] J. van Lunteren and T. Engbersen, "Fast and scalable packet classification," *JSAC*, 2003.

[9] Srinivasan Venkatachary Karthik Lakshminarayanan, Anand Rangarajan, "Algorithms for advanced packet classification with ternary cams," in *SIGCOMM*, 2005.

[10] F.Yu and R.H. Katz, "Efficient multi-match packet classification with TCAM," in *HOTI*, 2004.

[11] Y.-K. Kim, C.-C. Su, and Y.-C. Lin, "Efficient gray code based range encoding schemes for packet classification in TCAM," in *Globecom*, 2007, pp. 1834–1839.

[12] D. Taylor E.Spitznagel and J. Turner, "Packet classification using extended TCAMs," in *ICNP*, 2003.

[13] H. Che, Z.J. Wang, K. Zheng, and B. Liu, "DRES: Dynamic range encoding scheme for TCAM coprocessors," *IEEE Transaction on Computers*, vol. 57, no. 6, 2008.

[14] R. P. Draves, C. King, S. Venkatachary, and B. D. Zill, "Constructing optimal ip routing tables," in *INFOCOM 99*, March 1999.

[15] Q. Dong, S. Banerjee, J. Wang, D. Agrawal, and A. Shukla, "Packet classifiers in ternary cams can be smaller," in *SIGMETRICS*. 2006, pp. 311–322, ACM.

[16] C. R. Meiners, A. X. Liu, and E. Torng, "TCAM razor: A systematic approach towards minimizing packet classifiers in TCAMs," in *ICNP*, 2007.

[17] S. Suri, T. Sandholm, and P. R. Warkhede, "Compressing two-dimensional routing tables," *Algorithmica*, vol. 35, no. 4, pp. 287–300, 2003.

[18] C. Meiners, A. Liu, and E. Torng, "Topological transformation approaches to optimizing TCAM-based packet processing systems," in *SIGCOMM*, 2008.

[19] Chad R. Meiners, Alex X. Liu, and Eric Torng, "Bit weaving: A non-prefix approach to compressing packet classifiers in tcams," in *ICNP*, 2009, pp. 93–102.

[20] Alex X. Liu, Chad R. Meiners, and Yun Zhou, "All-match based complete redundancy removal for packet classifiers in TCAMs," in *Proceedings of the 27th Annual IEEE Conference on Computer Communications (Infocom)*, Phoenix, Arizona, April 2008.

[21] Alex X. Liu, Eric Torng, and Chad Meiners, "Firewall compressor: An algorithm for minimizing firewall policies," in *Proceedings of the 27th Annual IEEE Conference on Computer Communications (Infocom)*, Phoenix, Arizona, April 2008.

[22] Alex X. Liu and Mohamed G. Gouda, "Complete redundancy detection in firewalls," in *Proceedings of the 19th Annual IFIP Conference on Data and Applications Security, LNCS 3654*, Connecticut, US, August 2005, pp. 196–209.

[23] Alex X. Liu and Mohamed G. Gouda, "Complete redundancy removal for packet classifiers in tcams," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 4, pp. 424–437, 2010.

[24] Chad R. Meiners, Alex X. Liu, and Eric Torng, "Topological transformation approaches to optimizing tcam-based packet classification systems," in *SIGMETRICS/Performance*, 2009, pp. 73–84.

[25] Kai Zheng, Hao Che, Zhijun Wang, Bin Liu, and Xin Zhang, "DPPC-RE: TCAM-based distributed parallel packet classification with range encoding," *IEEE Trans. Computers*, vol. 55, no. 8, pp. 947–961, 2006.

[26] K. Zheng, C.C.Hu, H.B Lu, and B. Liu, "An ultra high throughput and power efficient tcam-based ip lookup engine," in *INFOCOM*, 2004.

[27] "Ports numbers," 2006, http://www.iana.org/assignments/port-numbers.

[28] Jonathan S. Turner David E. Taylor, "Classbench: A packet classification benchmark," in *IEEE INFOCOM*, 2005.

[29] David E. Taylor and Jonathan S. Turner, "Classbench: a packet classification benchmark," *IEEE/ACM Trans. Netw.*, vol. 15, no. 3, pp. 499–511, 2007.

11 Short Bios

- Anat Brelmer Barr received the B.Sc. degree in mathematics and computer science (Magna Cum Laude), the LL.B. degree in law, and the M.Sc. (Summa Cum Laude) and Ph.D. degrees (with distinction) in computer science from Tel Aviv University, Tel Aviv, Israel. In 2001, she confounded Riverhead Networks, a company that provides systems to protect from denial-of-service attacks. She was the Chief Scientist of the company, which was acquired by Cisco Systems in 2004. She joined the School of Computer Science at the Interdisciplinary Center, Herzliya, Israel, in 2004. Her research interests are in computer networks and distributed computing. Her current works are focused on designing routers and protocols that support efficient and reliable communication.

- Danny Hendler received the B.Sc. degree in mathematics and computer science and the M.Sc. and Ph.D. degrees in computer science from Tel Aviv University, Tel Aviv, Israel, in 1986, 1993, and 2004, respectively. He returned to Academy in 2001 after working for 18 years in the Israeli high-tech industry. He is currently an Assistant Professor with the Department of Computer Science, Ben-Gurion University of the Negev, Beer Sheva, Israel. Before joining Ben-Gurion University, he had been a Post-Doctoral Research Associate with the University of Toronto, Toronto, ON, Canada, and the Technion, Israel Institute of Technology, Haifa, Israel. His main research interests are distributed and parallel computation and packet classification.