Improved Behaviour of Tries by the "Symmetrization" of the Source

Yuriy A. Reznik [*]
RealNetworks, Inc.
2601 Elliott Avenue,
Seattle, WA 98121
yreznik@real.com

Wojciech Szpankowski[†]
Department of Computer Science
Purdue University
W. Lafayette, IN 47907
spa@cs.purdue.edu

Abstract

In this paper, we propose and study a pre-processing technique for improving performance of digital tree (trie)-based search algorithms under asymmetric memoryless sources. This technique (which we call a symmetrization of the source) bijectively maps the sequences of symbols from the original (asymmetric) source into symbols of an output alphabet resulting in a more uniform distribution. We introduce a criterion of efficiency for such a mapping, and demonstrate that a problem of finding an optimal for a given source (or universal) symmetrization transform is equivalent to a problem of constructing a minimum redundancy variable-length-to-block code for this source (or class of sources). Based on this result, we propose search algorithms that incorporate known (optimal for a given source and universal) variable-length-to-block codes and study their asymptotic behaviour. We complement our analysis with a description of an efficient algorithm for universal symmetrization of binary memoryless sources, and compare the performance of the resulting search structure with the standard tries.

1 Introduction

Digital trees (also known as *radix search trees*, or *tries*) represent a convenient technique for storing and retrieving data in computer's memory [10, 18]. In its original form, the trie is a data structure where a set of strings from an alphabet containing m symbols is stored in a form of an m-ary tree, where each string corresponds to a unique path. Typical applications of tries include searching and sorting, exact and approximate string matching, data compression schemes (see, e.g. [2, 18, 30]), etc.

It is well known, that the *average time of a successful search* in a trie containing n strings is asymptotically $\log(n)/h + O(1)$, while its *size* is asymptotically $n(1+\delta(n))/h + O(1)$, where h is the *entropy* of a stochastic process used to produce n input strings, and $\delta(n)$ is a fluctuation function of a small magnitude (cf. [18, 6, 11, 13, 25, 32, 14]).

The presence of the $1/h$ factor in the above formulas explains the *sensitivity* of tries to the parameters of the source. For example, if strings are produced by a *memoryless* source [4], its (per-symbol) entropy $h = -\sum_{i=1}^{m} p_i \log p_i$, where p_i $(i = 1, \ldots, m)$, are the probabilities of its symbols. It is easy to see, that when the source is *symmetric*, i.e. $p_i = 1/m$, its

[*]On leave from the Institute of Mathematical Machines and Systems, Kiev, Ukraine.

[†]This research was supported in part by NSF Grants NCR-9415491 and C-CR-9804760, and contract 1419991431A from sponsors of CERIAS at Purdue.

entropy attains the maximum (for m-ary sources) value $h = \log m$. However, if the source is *asymmetric*, i.e. $p_i \neq p_j$, for some indices $i \neq j$, or equivalently $p^* = \max_i \{p_i\} > 1/m$, its entropy can be arbitrary small: $\lim_{p^* \to 1} h = 0$, resulting in an unpredictable increase in both expected search time and expected size of these structures.

Even more affected by the asymmetry of the source are the *variance* and higher moments of the performance characteristics of tries. For example, it has been shown [32], that the variance of their depths (average number of steps in a successful search) in a memoryless model is asymptotically $\varsigma^2 \log n + O(1)$, where $\varsigma^2 = \left(h_2 - h^2\right)/h^3$, and $h_2 = \sum_{i=1}^{m} p_i \log^2 p_i$. If the source is symmetric, the factor $\varsigma = 0$, leaving the average magnitude of fluctuations in search times being only $O(1)$. However, if the source is asymmetric, the factor $\varsigma \neq 0$, which means that the standard deviation of search times in such tries is $O(\sqrt{\log n})$.

In an effort to improve the search time and/or memory requirements of tries, several modifications of the original trie structure have been proposed. Classical examples of such algorithms include *bucket tries* or b-tries [31, 18, 8, 23], *Patricia tries* [24], and *digital search trees* [3]. The last two implementations (Patricia and digital search trees) allowed the sizes of these structures to be independent from the statistics of the source (both have exactly n nodes). However, from the average search time perspective, all these modification offered improvements *only in an O(1) term* in the asymptotic expression, leaving the major term $\log n/h$ (and thus, their dependency on the parameters of the source) unchanged [18, 11, 20, 12, 22, 33].

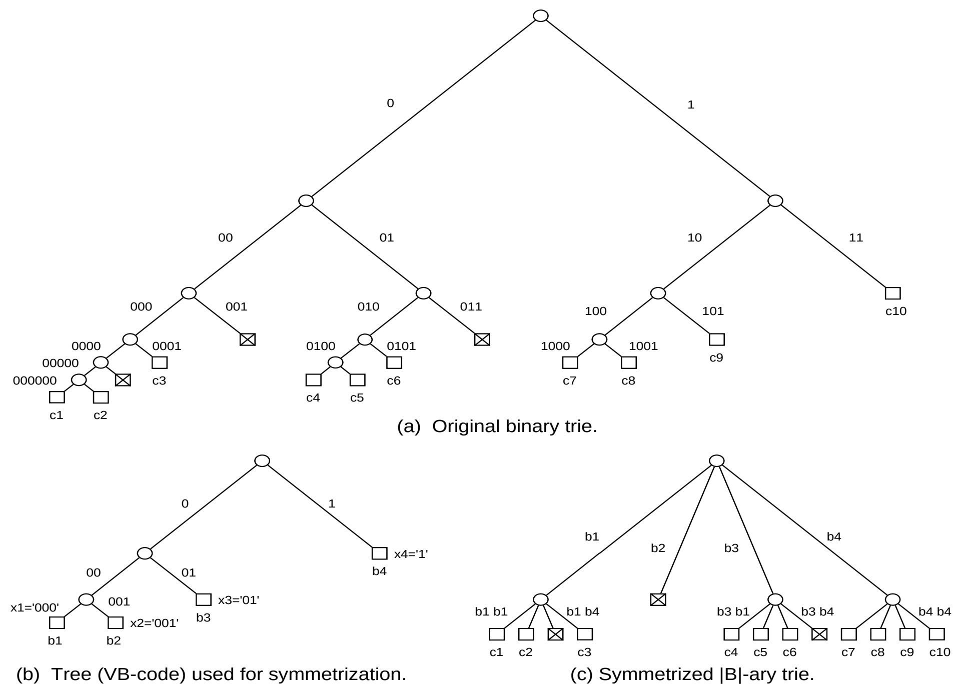
In this paper, we will show that substantially better results can be achieved by complementing the traditional trie search with a pre-processing technique, that "converts" symbols (or sequences of symbols) from the (asymmetric) source into ones that have almost uniform distribution. We call such a mapping a *symmetrization of the source*, and will demonstrate that it can be can realized by a specifically constructed *variable-length-to-block code* for a source (or class of sources).

To define an optimal construction of such codes, we introduce a criterion of their efficiency, and establish a connection between this quantity and the *redundancy* of variable-length-to-block codes. It turns out, that *both symmetrization and compression (minimization of code redundancy) of a memoryless source can be accomplished by the same code*. Such codes can be constructed with or without knowledge of the probabilities of the source.

For example, if the probabilities of the source are known, we will show that there exists a code, such that the average depth of the symmetrized trie is asymptotically upper bounded by $\log(n)/\log|t|\,(1 + h_{-\infty}/\log|t|\,(1 + o(1)))$, where $h_{-\infty} = -\log p_{\min}$, $p_{\min} = \min_i \{p_i\}$, and $|t|$ is the size of the code. It is clear, that by making $|t|$ large, the sensitivity of the resulting trie on the asymmetry of the source (parameter $h_{-\infty}$) can be made arbitrary small.

To demonstrate the viability of our idea in practice, we develop a symmetrization procedure based on a classic *universal variable-length-to-block code for memoryless sources* due to Trofimov [35, 19] and Lawrence [21]. This code is extremely easy to implement, and its overhead can be reduced to less than one lookup per input symbol. We compare the efficiency of a symmetrized trie based on this algorithm with the standard tries and show that it achieves better space- and time- performance characteristics under asymmetric sources.

This paper is organized as follows. In the next section, we will describe the structure of our algorithm, provide the necessary definitions and present our main results. The sketches of proofs will be provided in Section 3. In Section 4, we will explain the construction of a universal VB-code suitable for symmetrization, and will compare the performance characteristics of the resulting algorithm with ones achievable by the standard tries. Finally, in our concluding remarks, we emphasize the limitations of our present work and show directions



(a) Original binary trie.

(b) Tree (VB-code) used for symmetrization.

(c) Symmetrized |B|-ary trie.

Figure 1: A trie built from 10 binary strings: *c1=000000, c2=000001, c3=0001, c4=01000, c5=01001, c6=0101, c7=1000, c8=1001, c9=101, c10=11*; VB-code used for symmetrization; and the resulting symmetrized trie.

for future research.

2 Definitions and Main Results

Consider a set $C = \{c_1, \ldots, c_n\}$ of n (possibly *infinite*) sequences of symbols from an alphabet $A = \{a_1, \ldots, a_m\}$. A *trie* $T(C)$ over C can be constructed recursively as follows. If $n = 0$, the trie is an *empty external node*. If $n = 1$ (i.e. C has only one string), the trie is an *external node* containing a pointer to this single string in C. If $n > 1$, the trie is an *internal node* containing m pointers to the child tries: $T(C_1), \ldots, T(C_m)$, where each set C_i $(1 \leqslant i \leqslant m)$ contains suffixes of all strings from C that begin with a corresponding first symbol. For example, if a string $c_j = e_j f_j$ (e_j is a first symbol, and f_j is a string containing the remaining symbols of c_j), and $e_j = a_i$, then the string f_j will go into C_i. Thus, after all child tries $T(C_1), \ldots, T(C_m)$ are recursively processed, we arrive at a tree-like data structure, where the original strings $C = \{c_1, \ldots, c_n\}$ can be uniquely identified by the paths from the root node to non-empty external nodes (see Fig. 1.a).

A *variable-length-to-block code* (or VB-code) for sequences of symbols from A is defined by a *prefix free set* $X = \{x_1, \ldots, x_l\}$ of words over A, and a mapping[1]: $f : X \to B$, where $B = \{b_1, \ldots, b_l\}$ is the output alphabet. We present the structure of such code in a form of a tree t, where each path corresponds to a word in X and each leaf (external node) is labeled by a symbol from B (see Fig. 1.b). A coding algorithm starts from the root of the

[1]In coding theory this mapping is typically defined as $f : X \to B^r$, where $B = \{0, 1\}$ and $r \geqslant \log_2 |X|$. In our case, however, we don't have to use bits on the output, so we use a simpler mapping with $r = 1$ and $|B| = |X|$.

tree t and walks down according to the symbols in the input sequence until it reaches an external node, corresponding to a full word $x_j \in X$. It then picks the corresponding symbol b_j, sends it to the output, and begins the next search from the root of t.

In this paper, we propose a search technique that uses the above described VB-coding algorithm to *pre-process input sequences* before they are inserted in a trie. In other words, we construct a trie $T(t(C))$ where by $t(C)$ we denote a set of intermediate strings produced by a VB-code t (see Fig. 1.c). Our main goal is to find a code t improving characteristics of the trie $T(t(C))$ compared to a trie $T(C)$ built from the original set of strings C.

We settle this problem in a memoryless model. That is, we assume that input strings C are generated by a memoryless (or Bernoulli) [4] source S, which produces symbols from the alphabet A with probabilities $P = \{p_1, \dots, p_m\}$, $0 < p_i < 1$, $i = 1, \dots, m$, $\sum_{i=1}^{m} p_i = 1$. We use $|S|$ to denote the cardinality of the alphabet used by the source $|S| = |A|$, and $h(S)$ to denote its *entropy*:

$$h(S) = -\sum_{i=1}^{|S|} p_i \log p_i. \tag{1}$$

Consider now the distribution of symbols in strings $t(C)$ on the output of the VB code t. Clearly, in a memoryless model:

$$q_i := \Pr(b_i) = \Pr(x_i) = p_1^{r_1(x_i)} \dots p_m^{r_m(x_i)}, \tag{2}$$

where $r_j(x_i)$ denotes the number of occurrences of symbol a_j in a word x_i. Hence, we can say that strings $t(C)$ are produced by a memoryless source $t(S)$ with alphabet B and probabilities $Q = \{q_1, \dots, q_l\}$ (2).

Now, we are ready to introduce the main criterion that we will use for the selection of a VB-code for trie-based search algorithm.

Given a stochastic source S, we define its *asymmetry* $\sigma(S)$ as:

$$\sigma(S) = \frac{\log |S|}{h(S)} - 1. \tag{3}$$

Note, that unlike the entropy, the asymmetry of the source does not depend on the cardinality of its alphabet. It is also a unitless quantity (it does not depend on the bases of logarithms being used). The asymmetry is zero when the source is symmetric, and it grows to infinity when the source is asymmetric.

Using this criterion, we say that a code t achieves *symmetrization* of a given source S, if:

$$\sigma(t(S)) < \sigma(S).$$

If Θ represents a class of trees (codes) t that can be used for symmetrization, and

$$\sigma(t(S)) = \inf_{\theta \in \Theta} \sigma(\theta(S)), \tag{4}$$

then, we say, that the code t is *optimal* (within Θ) for symmetrization of the source S.

Moreover, if there exists a sequence of optimal trees $t_1, t_2, \dots$, corresponding to the nested classes $\Theta_1 \subset \Theta_2 \subset \dots$, such that for any (memoryless) source S:

$$\lim_{i \to \infty} \sigma(t_i(S)) = 0, \tag{5}$$

then the symmetrization achieved by algorithm producing codes t_i is *universal* (for a class of memoryless sources).

All these definitions have almost identical equivalents in the traditional coding theory (cf. [9, 5, 19]). The key difference between these approaches is that that the classic *VB codes* are designed to minimize the *redundancy* $R(t, S)$ (i.e. the difference between the per-symbol code length $C(t, S)$ and the entropy of the source $h(S)$) of their output, instead of making it symmetric.

Our first result, however, establishes a striking connection between these two criteria.

Lemma 1 *For any memoryless source S and VB-code t:*

$$R\left(t, S\right) \geqslant h(S)\, \sigma\left(t(S)\right). \tag{6}$$

This lemma immediately implies that VB-codes that are optimal and universal in a sense of minimizing redundancy, are automatically optimal and universal in achieving symmetrization of the output distribution.

So, our remaining statements are simply the results of applications of the existing VB-codes in the context of a trie-based search. We will be mostly interested in two characteristics of tries: their *average depth* D_n, representing the average number of lookups during a successful search, and the *average number of nodes* A_n, representing their size in computer's memory.

Theorem 1 *The average depth and the average number of nodes in a trie $T(t(C))$ constructed from n strings $t(C)$ originally produced by a memoryless source S and symmetrized by a VB-code t, are asymptotically (for large n):*

$$D_n(t) = \frac{\log n}{\log |t|}\left(1 + \sigma\left(t(S)\right)\right) + O(1), \tag{7}$$

$$A_n(t) = \frac{\log e}{\log |t|}\, n\left(1 + \delta(n)\right)\left(1 + \sigma\left(t(S)\right)\right) + O(1), \tag{8}$$

where $|t|$ - is the cardinality of the output alphabet used by the VB code, and $\delta(n)$ is a fluctuation function of a small magnitude.

Theorem 2 *There exists a code t^S, such that for a given source S:*

$$D_n\left(t^S\right) \leqslant \frac{\log n}{\log |t^S|}\left(1 + \frac{h_{-\infty}(S)}{\log |t^S|}\left(1 + o(1)\right)\right), \tag{9}$$

$$A_n\left(t^S\right) \leqslant \frac{\log e}{\log |t^S|}\, n\left(1 + \delta(n)\right)\left(1 + \frac{h_{-\infty}(S)}{\log |t^S|}\left(1 + o(1)\right)\right), \tag{10}$$

provided that n and $\left|t^S\right|$ are large, and $h_{-\infty}(S) = -\log p_{\min}$, where $p_{\min} = \min_i\{p_i\}$ is the probability of the least likely symbol generated by the source S.

Theorem 3 *There exists a code t^*, such that for any memoryless source:*

$$D_n\left(t^*\right) \leqslant \frac{\log n}{\log |t^*|}\left(1 + \frac{3}{2}\frac{\log\log |t^*|}{\log |t^*|}\left(1 + o(1)\right)\right), \tag{11}$$

$$A_n\left(t^*\right) \leqslant \frac{\log e}{\log |t^*|}\, n\left(1 + \delta(n)\right)\left(1 + \frac{3}{2}\frac{\log\log |t^*|}{\log |t^*|}\left(1 + o(1)\right)\right), \tag{12}$$

provided that n and $|t^*|$ are large.

In other words, we can conclude that using a sufficiently large symmetrizing VB-code t, we can make the sensitivity of both search time and size of tries on the asymmetry of the source arbitrary small. This result can be achieved with or without *a priori* knowledge of the parameters of the source.

3 Sketches of Proofs

First, we need to bring several standard definitions from the source coding theory. We say that the *average delay* $d(t,S)$ of a VB-code t for a memoryless source S is a quantity [19]:

$$d(t,S) = \sum_{i=1}^{|X|} \Pr(x_i)\,|x_i|\,, \tag{13}$$

where $x_i \in X$ are strings corresponding to the paths from root to external nodes in the tree t ($|t| = |X|$), $\Pr(x_i)$ are their probabilities (2), and $|x_i|$ are their lengths.

The *cost* $C(t,S)$ of a code t for S is the average number of units of information (e.g. bits) produced by this code per each input symbol. Thus, the cost of a VB-code t is

$$C(t,S) = \frac{\lceil \log |t| \rceil}{d(t,S)}, \tag{14}$$

where $\lceil \log |t| \rceil$ represents a number of output units sufficient to encode paths in tree t.

The redundancy $R(t,S)$ of a VB-code t is the difference:

$$R\,(t,S) = C(t,S) - h(S) = \frac{\lceil \log |t| \rceil}{d(t,S)} - h(S), \tag{15}$$

The claim of Lemma 1 follows directly from our definition of the asymmetry of the source (3), formula (15), and an identity (cf. Sidelnikov [29], Jelinek and Schneider [16, Lemma 3], and Krichevsky [19, Lemma 2.2.1]):

$$h(S) = -\frac{1}{d(t,S)} \sum_{i=1}^{|X|} \Pr(x_i)\,\log \Pr(x_i) = \frac{h\,(t(S))}{d(t,S)}. \tag{16}$$

To prove our Theorem 1, it is sufficient to use known asymptotic expressions for the average depth D_n and the average number of internal nodes A_n in a trie constructed in a memoryless model:

$$\begin{aligned} D_n &= \frac{1}{h(S)} \log n + O(1), \\ A_n &= \frac{\log e}{h(S)}\, n\,(1 + \delta(n)) + O\,(1)\,, \end{aligned}$$

where $\delta(n)$ is a fluctuation function of a small magnitude (cf. [18, 6, 11, 13, 25, 32]).

Replacing $h(S)$ in the above expressions with $h(t(S))$, and using (3) we arrive at formulas (7) and (8) claimed by the Theorem 1.

To prove our Theorem 2 we will need to use a redundancy bound for an *optimal VB-code* t^S for a memoryless source S (cf. Khodak [17] and Jelinek and Schneider [16]):

$$R\left(t^S, S\right) \leqslant \frac{h_{-\infty}(S)}{d\left(t^S, S\right)}, \tag{17}$$

where $h_{-\infty}(S) = -\log p_{\min}$, and $p_{\min} = \min_i \{p_i\}$ is the probability of the least likely symbol in S.

From (17) it follows, that $\lim_{d(t^S,S)\to\infty} R\left(t^S, S\right) = 0$, so we can rewrite (15) in a form

$$\log \left|t^S\right| = h(S) d\left(t^S, S\right)(1 + o(1))\,. \tag{18}$$

Using our lemma (6)[2], redundancy bound (17), and subsequently (18), we arrive at:

$$\sigma\left(t^S(S)\right) \leqslant \frac{1}{h(S)} \frac{h_{-\infty}(S)}{d\left(t^S, S\right)} = \frac{h_{-\infty}(S)}{\log|t^S|}\left(1+o(1)\right),$$

which, combined with (7) and (8) leads to the formulas (9) and (10) in Theorem 2.

Finally, to prove our Theorem 3, we will use the redundancy bound for a *universal VB-code t^* for memoryless sources* due to Trofimov [35] (see also Krichevsky [19]):

$$R\left(t^S, S\right) \leqslant \frac{3}{2} \frac{\log d\left(t^*, S\right) + C}{d\left(t^*, S\right)}, \tag{19}$$

where C is a constant.

Using the same arguments as above (most notably (6) and (18)), we can show, that the asymmetry of the output symbols in such code:

$$\sigma\left(t^*(S)\right) \leqslant \frac{3}{2} \frac{\log\log|t^*|}{\log|t^*|}\left(1+o(1)\right),$$

resulting in formulas (11) and (12) claimed by the Theorem 3.

4 Implementation and Performance Comparisons

In this paper we propose an algorithm that pre-processes (symmetrizes) input sequences in a trie-based search, such that the resulting structure becomes less sensitive to the asymmetry of the source. It is immediately clear, however, that in order for such a technique to be successful, the *complexity of symmetrization* must be relatively small compared to the complexity of the trie search. So in practice, we have to restrict our choice of algorithms to ones that can be implemented using not more than one lookup per input symbol.

This automatically means, that algorithms that construct VB-codes adaptively, based on the prior statistics of the source, are not likely candidates for such a purpose. In fact, we will have to dismiss all adaptive algorithms due to an even simpler argument: in our search algorithm we associate paths in the code tree t with unique symbols in a symmetrized trie $T(t)$. Therefore, any change in the structure of tree t will also require reconstruction of the entire search structure T.

For these reasons, in our implementation we decided to use a *universal variable-length-to-block-code for memoryless sources*, which was developed in late 1970s by V. Trofimov [35] (see also [19]) and (independently) by J. C. Lawrence [21].

For simplicity, we demonstrate the construction of this code for binary sources only. We use p and $q = 1-p$ to designate the probabilities of symbols 0 and 1 correspondingly.

Consider a binary sequence x of length $|x|$, containing $r_0(x)$ zeros. Fix some positive integer number z. We say, that the sequence x reaches a leaf in the code tree t^*_z, if

$$\log_2 \binom{|x|}{r_0(x)} \leqslant z < \left[\begin{array}{ll} \log \binom{|x|+1}{r_0(x)+1} & \text{if } r_0(x) \leqslant \frac{|x|}{2}, \\ \log \binom{|x|+1}{r_0(x)} & \text{if } r_0(x) > \frac{|x|}{2}. \end{array} \right. \tag{20}$$

The construction is recursive: start moving from the root, until conditions (20) are met. In Fig.2 we illustrate the results of construction of such codes for parameters $z = 1, 2, 3$.

[2]Note, that since the cardinality of our output alphabet always matches the number of leaves, the ceiling of the logarithm in (15) has no effect, so that formula (6) turns into equality.

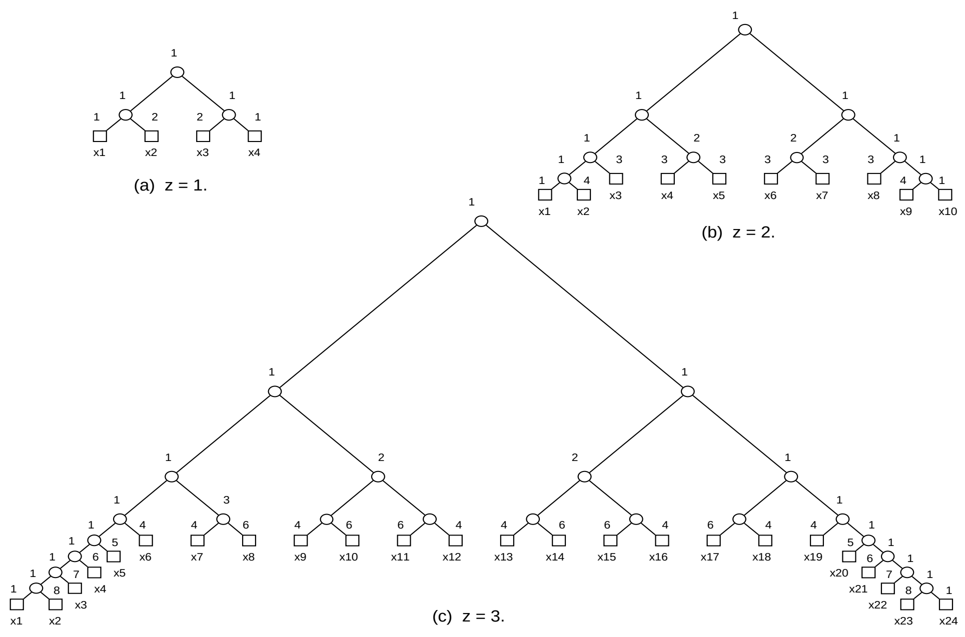


Figure 2: Universal variable-length-to-block codes t_z^* constructed using $z = 1, 2, 3$. Nodes are labeled by the corresponding values of $\binom{|x|}{r_0(x)}$.

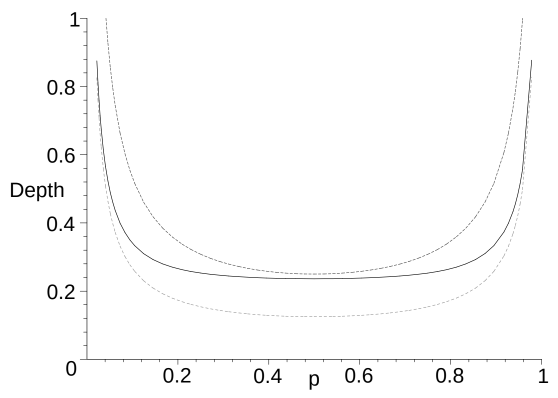
Observe that since such trees have fixed structures, the corresponding encoding/decoding algorithms can be implemented in a very efficient manner. One such implementation, can employ lookup tables to process multiple levels of such trees at a time. For instance, to implement a scanning procedure for a tree t_3^* (see Fig.2.c), it is sufficient to create a single table containing $2^{2^3} = 256$ entries, corresponding to all possible extensions of this code to 8 bits. The selection of a leaf in this case can be done using only one lookup.

We demonstrate the efficiency of the symmetrization achieved by the universal VB-codes in Fig.3. Horizontal axis in this plot corresponds to the probability of symbol 0 in binary alphabet. Vertical axis corresponds to the asymmetry $\sigma(t_z^*(S))$ of the output distributions for universal VB codes with parameters $z = 1, 2, 3, 4,$ and 8. We can observe that the higher is the parameter z, the flatter is the output distribution.

To assess the efficiency of a trie symmetrized by a code t_3^* (see Fig.2.c) we compare it with tries constructed using fixed-length blocks of input bits as symbols of their alphabet. Observe, that the shortest path in code t_3^* has a length of 4, while the longest path has a length of 8. So, we will construct three tries: $T(t_3^*(S))$ a symmetrized trie, $T(S^4)$ a trie built using 4-bits blocks, and $T(S^8)$ a trie built using 8-bits blocks.

We plot the factors of the $O(\log n)$ term in the asymptotic expressions for the depths of these tries in Fig.4. As expected, the trie $T(S^4)$ demonstrates similar performance to $T(t_3^*(S))$ when the source is almost symmetric $p \sim 1/2$, but quickly becomes slower when $p \to 0$ or $p \to 1$. The trie $T(S^8)$, on the other hand is almost 2 times faster than $T(t_3^*(S))$ when $p \sim 1/2$, but quickly slows down to the $T(t_3^*(S))$ speed when source is asymmetric.

Considering the factors of the $O(n)$ terms in the expressions for sizes of these tries (Fig.4), we find, that the trie $T(S^8)$ is extremely large, being about 6 times of the size of $T(t_3^*(S))$ when the source is symmetric, and further increasing the gap in the asymmetric case. On the other hand, the tree $T(S^4)$ is slightly more compact in the central range, but it matches the size of $T(t_3^*(S))$ around $p = 0.1$ and $p = 0.9$, and grows even faster beyond that points.



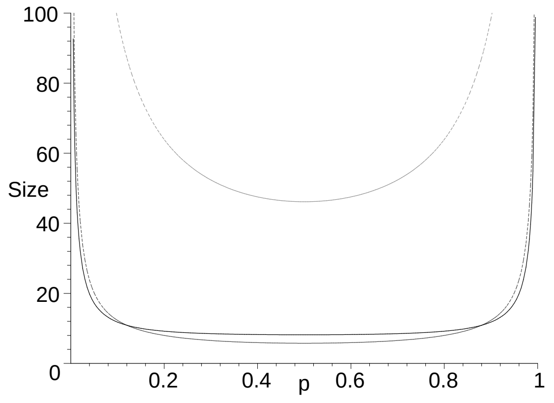


Figure 3: Factors of the expected depths (left) and the expected sizes (right) of tries: $T(t_3^*(S))$ (solid line), $T(S^4)$ (dashed), and $T(S^8)$ (dotted).

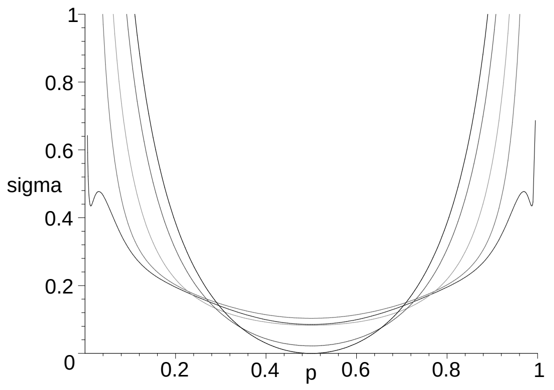


Figure 4: The asymmetry $\sigma(t_z^*(S))$ of the output distributions of universal VB codes with parameters $z = 1, 2, 3, 4,$ and 8. Flatter curves correspond to higher values of z.

Overall, we can see that the symmetrized trie $T(t(S))$ achieves a very desirable, from practical point of view, balance in time- and space- performance, making it a much more efficient data structure for sources with high (or unknown) asymmetry.

5 Concluding Remarks

In this paper we have shown that the performance of tries under asymmetric memoryless sources can be substantially improved using the symmetrization of the source. The symmetrization can be effectively achieved using the standard VB codes for memoryless sources.

The extension of our results for Markovian sources should be an interesting future task. The relevant tools from coding theory in this case are universal codes for Markovian sources of fixed order [34], and so-called *twice-universal codes* [28] when the order is not known.

Another possible direction for future research is to use symmetrization to improve the performance of tries with *adaptive branching* (cf. [1, 26, 7, 27]). These are relatively new algorithms, which execute searches in $O(\log^* n)$ (or even $O(1)$ [26]) steps when the source is symmetric, but only $O(\log \log n)$-fast in the asymmetric case. The use of symmetrization in such a context is likely to result in very substantial (affecting order of n) improvements.

References

[1] A. Andersson and S. Nilsson, Improved Behaviour of Tries by Adaptive Branching, *Information Processing Letters*, **46** (1993) 295–300.

[2] A. Aho, J. Hopcroft, J. Ullman, The Design and Analysis of Computer Algorithms (Addison-Wesley, Reading MA, 1974)

[3] E. G. Coffman, Jr. and J. Eve, File Structures Using Hashing Functions, *Comm. ACM*, 13 (7) (1970) 427–436.

[4] T. M. Cover and J. M. Thomas, *Elements of Information Theory*, (Wiley, New York, 1991).

[5] L. D. Davisson, Universal Noiseless Coding, *IEEE Trans. Inform. Theory*, 19 (6) (1973) 783–795.

[6] L. Devroye, A Note on the Average Depths in Tries, *SIAM J. Computing*, 28 (1982) 367–371.

[7] L. Devroye, Analysis of Random LC Tries, *Rand. Structures & Algorithms*, 19 (3) (2001) 359–375.

[8] R. Fagin, J. Nievergelt, N. Pipinger, H. R. Strong, Extendible Hashing – A Fast Access Method for Dynamic Files, *ACM Trans. Database Syst.*, 4 (3) (1979) 315–344.

[9] B. M. Fitingof, Optimal Coding in the Case of Unknown and Changing Message Statistics, *Probl. Inform. Transm.*, 2, (2) (1965) 3–11 (in Russian) 1–7 (English Transl.)

[10] E. Fredkin, Trie Memory, *Comm. ACM*, 3 (1960) 490–500.

[11] P. Flajolet and R. Sedgewick, Digital Search Trees Revisited, *SIAM J. Computing*, 15 (1986) 748–767.

[12] P. Flajolet and B. Richmond, Generalized Digital Trees and Their Difference-Differential Equations, Random Structures and Algorithms, 3 (1992) 305-320.

[13] P. Jacquet and M. Régnier, Trie Partitioning Process: Limiting Distributions, *Lecture Notes in Computer Science*, 214 (Springer-Verlag, New York, 1986) 196–210.

[14] P. Jacquet, and W. Szpankowski, Analysis of digital tries with Markovian dependency, *IEEE Trans. Inform. Theory*, 37 (1991) 1470–1475.

[15] P. Jacquet and W. Szpankowski, Autocorrelation on Words and Its Applications. Analysis of Suffix Trees by String-Ruler Approach, *J. Combin. Theory Ser. A*, 66, (1994) 237–269.

[16] F. Jelinek, and K. S. Schneider, On Variable-Length-to-Block Coding, *IEEE Trans. Inform. Theory*, 18 (6) (1972) 765–774.

[17] G. L. Khodak, Redundancy Estimates for Word-Based Codes for Bernoulli Sequences, *Probl. Inform. Trans.*, 8 (2) (1972) 21–32 (in Russian).

[18] D. Knuth, *The Art of Computer Programming. Sorting and Searching. Vol. 3* (Addison-Wesley, Reading MA, 1973).

[19] R. E. Krichevsky, Universal Data Compression and Retrieval. (Kluwer, Norwell, MA, 1993).

[20] P. Kirschenhofer and H. Prodinger, Some further results on digital search trees, *Lecture Notes in Computer Science*, 229 (Springer-Verlag, New York, 1986) 177–185.

[21] J. C. Lawrence, A New Universal Coding Scheme for the Binary Memoryless Source, *IEEE Trans. Inform. Theory*, 23 (4) (1977) 466–472.

[22] G. Louchard, Digital Search Trees Revisited, Cahiers du CERO, 36 (1995) 259-27.

[23] H. Mahmoud, P. Flajolet, P. Jacquet and M. Régnier, Analytic Variations on Bucket Selection and Sorting, *Acta Informatica* 36 (2000) 735–760.

[24] D. A. Morrison, PATRICIA – Practical Algorithm To Retrieve Information Coded in Alphanumeric, *J. ACM*, 15 (4) (1968) 514–534.

[25] B. Pittel, Paths in a Random Digital Tree: Limiting Distributions, *Advances in Applied Probability*, 18 (1986) 139–155.

[26] Yu. A. Reznik, Some Results on Tries with Adaptive Branching, *Lecture Notes in Computer Science*, 1858 (Springer-Verlag, New York, 2000) 148–158.

[27] Yu. A. Reznik, On the Average Density and Selectivity of Nodes in Multi-Digit Tries, *ACM Symposium on Theory of Computing (STOC'02)* (2002) – submitted.

[28] B. Ya. Ryabko, Twice Universal Coding, *Probl. Inform. Trans.*, 20 (3) (1984) 24–28 (in Russian).

[29] V. M. Sidelnikov, On Statistical Properties of Transformations Carried out by Finite Automata, *Cybernetics*, 6 (1965) 1–14 (in Russian)

[30] J. Storer, Data Compression: Methods and Theory, (Computer Science Press, Rockville, MD, 1988).

[31] E. H. Sussenguth, Jr., Use of Tree Structures for Processing Files, *Comm. ACM*, 6 (5) (1963) 272–279.

[32] W. Szpankowski, Some results on V-ary asymmetric tries, *J. Algorithms*, 9 (1988) 224–244.

[33] W. Szpankowski, Patricia tries again revisited, *J. ACM*, 37 (1990) 691–711.

[34] V. K. Trofimov, The Redundancy of the Universal Encoding of Arbitrary Markov Sources, *Probl. Inform. Trans.*, 10 (4) (1974) 16–24 (in Russian).

[35] V. K. Trofimov, Universal Variable-Length to Block Codes for Bernoulli Sources, *Methods of Discrete Analysis*, 29 (Novosibirsk, 1976) 87–100 (in Russian)