Using T-Codes as Locally Decodable Source Codes

Ulrich Speidel
Dept. of Computer Science
The University of Auckland
Email: ulrich@cs.auckland.ac.nz

T. Aaron Gulliver
Dept. of Electrical and Computer Engineering
The University of Victoria
Email: agullive@ece.uvic.ca

Ali Makhdoumi, Muriel Médard
EECS
Massachusetts Institute of Technology
Email: {makhdoum, medard}@mit.edu

***Abstract*—A locally decodable source code (LDSC) allows the recovery of arbitrary parts of an unencoded message from its encoded version, using only a part of the encoded message as input, a challenge that arises when searching within compressed data sets. Simple source codes such as Huffman codes or Lempel-Ziv compression are not well suited to this task: A decoder starting at an arbitrary point within the compressed sequence generally cannot determine its position with respect to the boundaries between encoded symbols, or requires information found before the starting point in order to be able to decode. In this paper, we propose the use of subsets of self-synchronising variable-length T-codes as source codes and show that local decoding is feasible and practical using subsets of T-codes with bounded synchronisation delay (BSD).**

I. Introduction

In source coding, one tries to compress data from a given source S, i.e., to express it with the lowest possible number of symbols from an alphabet A. Source codes are said to be *locally decodable* if they let a decoder recover source symbols encoded after an arbitrary position *inside* such a compressed message, without requiring decoding from the start. However, most source codes are not well suited to local decoding, e.g., because they rely on a dictionary conveyed before the local starting position, or because codeword boundaries cannot be determined locally.

Several authors [2], [3] have investigated causal source codes, which allow decoding of each source symbol from the compressed message without looking at subsequent codewords. The counterpart of LDSCs in channel coding are update efficient codes [4], where changing a source symbol requires only a small number of symbols in the compressed message to be updated.

The problem of efficiently reproducing symbols from a compressed representation of a source has also been studied in the data structure literature. For example, Bloom filters [5] are data structures for storing a set in a compressed form that allow membership queries to be answered in bounded time. The dictionary problem [6] and succinct data representation [7], [8] in the field of succinct data structures are further examples of problems involving both compression and the ability to recover a single symbol of the input message efficiently.

There are two main approaches to source coding: fixed-length source coding and variable-length source coding. Reference [1] introduces fixed-length LDSCs. Our paper proposes variable-length LDSCs in the form of subsets of Titchener's T-codes [11], [13] with BSD: We show that they can achieve compression for a variety of sources and provide a new bound on the expected synchronisation delay, and a modification to the matching algorithm described in [18], [19] to find suitable T-codes for local decoding.

In the case of variable-length source codes such as Huffman codes [9], the compressed message consists of a sequence of variable-length codewords which are in turn finite sequences with symbols from A. The main challenge in local decoding of such codes is to find a demonstrably valid codeword boundary after the starting position inside the compressed sequence. This requires the decoder to *synchronise* with the source code. However, variable length codes are not necessarily universally synchronisable, and this is generally also the case for Huffman codes. Similarly, synchronisable codes are not necessarily of interest in source coding: Comma-free codes provide no compression potential, for example, and statistically synchronisable codes with unbounded synchronisation delay cannot guarantee true local decodability. The latter also applies to T-codes, which cannot synchronise in some semi-infinite periodic sequences. However, these sequences consist of repeating periodic codewords and do not occur if one encodes with non-periodic T-code codewords only (see [10] and Section IV below). This limits the synchronisation delay while retaining most of the potential for compression.

In this paper, λ denotes the empty sequence, S denotes the set of source symbols and C the set of encoded symbols (codewords) over some alphabet A such that $C \subset A^+$, where $A^+ = \{(a_1, a_2, \ldots, a_q) | q \in \mathbb{N}^+, a_i \in A, 1 \leq i \leq q\}$, and $A^* = A^+ \cup \{\lambda\}$. A *variable length code* (VLC) consists of an *encoder* and a *decoder* implementing the mappings $f : S \rightarrow C$ and $g : C^+ \rightarrow S^+$, respectively, where $C^+ = \{(x_1, x_2, \ldots, x_m) | m \in \mathbb{N}^+, x_i \in C, 1 \leq i \leq m\}$ and $S^+ = \{(w_1, w_2, \ldots, w_\mu) | \mu \in \mathbb{N}^+, w_i \in S, 1 \leq i \leq \mu\}$. For example, a binary Huffman code encoding English letters A to Z into binary codewords might use $S = \{A, B, C, \ldots, Z\}$, $A = \{0, 1\}$ and, for the appropriate probabilites of symbols in S, we might obtain $C = \{00, 0100, \ldots, 1111\}$. Note further that as $C \subset A^+$, we also have $C^+ \subset A^+$.

The next section gives a formal definition of variable-length LDSCs, reviews T-codes and discusses the principles of source coding with T-codes, followed in Section III by a brief review of the basic matching algorithm that finds a T-code for a given source probability distribution. We then move to the decoding side in Section IV. It discusses how a T-code decoder can synchronise in the compressed message, gives a

simple expected synchronisation delay (ESD), and discusses how removing periodic codewords from the T-code results in BSD codes with guaranteed local decodability. This codeword removal necessitates changes to the basic matching algorithm, which we propose in Section V, followed by examples in Section VI and then our conclusion.

II. Background

A. Locally decodable variable length source codes

Consider a sequence $x = w_1 w_2 \dots w_\mu$ with μ codewords w_j from a VLC C over A. As C is a code over A, we may also read x as a sequence over A, i.e., as $x = x_1 x_2 \dots x_m$ with $x_i \in A$. Let the suffix $x_{i'} x_{i'+1} \dots x_m$ of x with $x_{i'}, x_{i'+1}, \dots, x_m \in A$ be the input to a decoder g for C. If there exists a $d \geq 0$ such that the decoder can identify $x_{i'+d} x_{i'+d+1} \dots x_m$ unambigously as the concatenation of the last j' codewords $w_{\mu-j'+1} w_{\mu-j'+2} \dots w_\mu$ in x, then g is said to have *synchronised* before the $i'+d$-th symbol. If $i'+d$ is the smallest possible value for a given i', then d is the *synchronisation delay* experienced by g.

If a finite d exists for any given finite i' and arbitrary w_j as $\mu \to \infty$, C is said to be *self-synchronising*. If a finite bound on d exists for a given C, then C is said to be a *bounded synchronisation delay* (BSD) code. If d is unbounded but finite with probability 1 in a semi-infinite random sequence x_i, then C is said to be *statistically self-synchronising*.

A VLC with BSD is *locally decodable* as it can decode part of a sequence from C^+ after starting at an arbitrary position i' inside the sequence. Note that $x_0 x_1 \dots x_{i'-1}$ must be in A^* but does not need to be in C^+, i.e., the start position does not have to coincide with the start of a codeword from C. Note that we regard g as unsynchronised until g can *deduce the correctness* of its decoded output $w_{\mu-j'+1} w_{\mu-j'+2} \dots w_\mu$ from $x_{i'} x_{i'+1} \dots x_m$. This is a more stringent requirement than mere (possibly coincidental) correctness of the output, and can be met if C is a T-code (see Section IV).

B. T-codes

T-codes are VLCs constructed from an alphabet A in i steps via a recursive copy and append process called *T-augmentation*. Starting with $A^{()}_{()} = A$ as a trivial T-code for $i = 0$, we define T-augmentation by the following recurrence relation:

$$A^{(k_1,k_2,\dots,k_{i+1})}_{(p_1,p_2,\dots,p_{i+1})} = \bigcup_{j=0}^{k_{i+1}} \{p_{i+1}^j s | s \in A^{(k_1,k_2,\dots,k_i)}_{(p_1,p_2,\dots,p_i)} \backslash \{p_{i+1}\}\} \cup \{p_{i+1}^{k_{i+1}+1}\} \quad (1)$$

where $A^{(k_1,k_2,\dots,k_i)}_{(p_1,p_2,\dots,p_i)}$ denotes the *T-code (set)* after i steps (said to be at *T-augmentation level* i), $p_{i+1} \in A^{(k_1,k_2,\dots,k_i)}_{(p_1,p_2,\dots,p_i)}$ is the *T-prefix* and $k_{i+1} \in \mathbb{N}^+$ the *T-expansion parameter* or *copy factor* for the T-augmentation from level i to level $i+1$. The sub- and superscript lists of T-prefixes and T-expansion parameters, $(p_1, p_2, \dots, p_i)$ and $(k_1, k_2, \dots, k_i)$, are called the *T-prescription* of $A^{(k_1,k_2,\dots,k_i)}_{(p_1,p_2,\dots,p_i)}$.

Example: Let $A = \{0, 1\}$. Select $p_1 = 0$ and $k_1 = 2$ to obtain $A^{(2)}_{(0)} = \{1, 01, 000, 001\}$. Now select $p_2 = 01$, $k_2 = 1$ to obtain $A^{(2,1)}_{(0,01)} = \{1, 000, 001, 011, 0101, 01000, 01001\}$, etc.

C. Source coding with T-codes

As variable-length codes, T-codes may be used as source codes, following the well-known principle of assigning short codewords to source symbols of high probability of occurrence and long codewords to source symbols of low probability. Let $\#A$ and $\#S$ denote the cardinalities of A and S, $P(s_j)$ the probability of occurrence of the j'th symbol $s_j \in S$, and let $f : S \to C$ be an encoding of the source symbols, such that $f(s_j) \in C$. The aim of simple source codes such as Huffman codes is to find a C and f that minimise the *redundancy* r:

$$r = \sum_{j=1}^{\#S} P(s_j) \left[|f(s_j)| + \log_{\#A} P(s_j) \right]. \quad (2)$$

Note that C and f are generally not unique as r does not depend on C itself, but merely on the code length distribution δ_C of C, which is generally shared by multiple VLCs. However, f always satisfies the following condition:

$$P(s_j) > P(s_{j'}) \implies |f(s_j)| \leq |f(s_{j'})|. \quad (3)$$

If minimal r is the only requirement, we may simply construct C as a Huffman code. If C is also to meet other criteria, e.g., to be a T-code, we must minimise r over the respective set of codes meeting these criteria. While every T-code is a potential outcome of Huffman's algorithm for certain source probability distributions, the converse does not apply: Most Huffman codes are not T-codes and so do not have the structural properties that give rise to self-synchronisation (see Section IV).

T-augmentation, on the other hand, pays no attention to coding efficiency: While the choice of the T-prefixes and T-expansion parameters lets us construct a wide range of sets with varying δ_C, the construction does not lend itself to minimising r [18]. However, as discussed in Section IV, it guarantees good synchronisation performance. Using T-codes thus implies a trade-off between self-synchronisation and efficiency.

The aim is then to find a T-code with minimal r for a given source probability distribution $P(S)$. Titchener [16] suggested to look for a "small" rather than minimal r as source statistics tend to be approximate. Higgie [15] presented a database of "best T-codes" minimising r for T-codes where $k_i = 1$ for all i, i.e., a subset of possible δ_C only.

The only known strategy for minimising r in the general case remains an exhaustive search of all feasible δ_C. This was first proposed by one of the authors in [14] and [18], with improvements in [19]. The next section describes the basic version of this algorithm.

III. The basic matching algorithm

The algorithm from [18], [19] operates as a breadth-first-search algorithm (BFS) using the well-known branch-and-bound technique: Each node in the search tree represents

a T-code, with the tree root representing A. Each branch originating from such a node represents a T-augmentation of the node's T-code with a particular feasible T-prefix and T-expansion parameter. To find the minimal r, we need, in principle, to visit each node as it appears at the front of the BFS queue, and:

1) If the corresponding T-code is large enough, calculate its redundancy r with respect to $P(S)$. Given its δ_C, assign codewords to source symbols in accordance with (3).
2) Create and enqueue any feasible child nodes according to the feasibility criteria discussed below. They ensure that the tree remains finite in size.
3) Dequeue each node once it has been processed.
4) Keep a record of the node with the lowest r that was found in the process.

We simplify this search by exploiting a number of shortcuts. These are described in part in [18] and, with the improvements presented here, in [19]. They include:

- Using codeword length distributions δ_C rather than actual T-codes as nodes in the tree: r is a function of δ_C only, and for $C = A^{(k_1,k_2,\ldots,k_{i+1})}_{(p_1,p_2,\ldots,p_{i+1})}$, δ_C depends only on $|p_{i+1}|$ but not on the specific choice of p_{i+1} among the codewords of length $|p_{i+1}|$ in $A^{(k_1,k_2,\ldots,k_i)}_{(p_1,p_2,\ldots,p_i)}$. Instead of T-augmenting multiple T-codes with the same δ_C to compute their respective r, we can instead obtain δ_C by "virtual T-augmentation" (VTA) of the codeword length distribution of $A^{(k_1,k_2,\ldots,k_i)}_{(p_1,p_2,\ldots,p_i)}$ with T-prefix length $|p_{i+1}|$. A single VTA can thus account for all T-codes with the same codeword length distribution and all candidate T-prefixes of length $|p_{i+1}|$.
- Using only T-prescriptions in anti-canonical form [12], [17], [18], i.e., $k_i + 1$ must always be prime.
- Using non-decreasing T-prefix lengths in subsequent VTAs.
- Ignoring any codewords longer than $\lceil (\#S-1)/(\#A-1) \rceil$ (see [18], 10.4.7).

When we process a node with distribution δ_C, a further VTA with candidate T-prefix length $|p|$ and candidate T-expansion parameter k is only feasible under the following conditions:

- Let L be the length of the longest codeword presently assigned. If δ_C is large enough to encode S, the VTA must create at least *two* new codewords shorter than L to obtain a redundancy gain from reassigning a source symbol currently encoded with length L or less *and* compensate for the loss of the T-prefix. We thus require $k|p| + \ell_2 < L$ where ℓ_2 is the smallest length for which δ_C contains at least two codewords not longer than ℓ_2.
- $k|p|+\ell_2 \leq \lceil (\#S-1)/(\#A-1) \rceil$. This condition ensures that VTAs create codewords within a length range of interest: Since the maximum depth Huffman code for S is $\lceil (\#S-1)/(\#A-1) \rceil$ deep and simultaneously a T-code, we cannot encode S more efficiently with longer codewords. Feasible VTAs must create new codewords at or below this length. Note that δ_C large enough to encode the source implicitly meet this criterion via $k|p|+\ell_2 < L$.
- The *residual redundancy* (partial sum of (2) including those terms with encodings shorter than $|p|$), must not exceed the lowest redundancy found thus far. We may also abort the investigation of any δ_C as soon as the redundancy exceeds this value.

In combination, these criteria limit the search space.

IV. T-code self-synchronisation and expected synchronisation delay

Consider a compressed message x in the form of a concatenation of codewords from $A^{(k_1,k_2,\ldots,k_i)}_{(p_1,p_2,\ldots,p_i)}$. The boundary between two codewords in this concatenation is called an *i-boundary*. A T-code decoder that has identified an i-boundary in x can henceforth identify the correct codewords in $A^{(k_1,k_2,\ldots,k_i)}_{(p_1,p_2,\ldots,p_i)}$, and we say that it is *synchronised* to level i.

We may thus model the synchronisation process as a Markov chain, whose states represent the highest T-augmentation level i for which the decoder has been able to identify an i-boundary. Note that by (reffulltaug), every i-boundary is also an $(i-1)$-boundary. A decoder that identifies an $(i-1)$-boundary as also being an i-boundary thus synchronises from state (level) $i-1$ to state (level) i. An unsynchronised decoder starts in state 0, meaning it knows the location of a (trivially identified) 0-boundary in x. As it synchronises, the decoder progressively identifies 1-, 2-, and eventually n-boundaries.

The transition criterion for the chain is based on the insight that a decoder, upon reaching state $i-1$, can henceforth correctly identify the codewords from $A^{(k_1,k_2,\ldots,k_{i-1})}_{(p_1,p_2,\ldots,p_{i-1})}$ in the remainder of x. These may include the T-prefix p_i. By (1), mere $(i-1)$-boundaries inside a codeword from $A^{(k_1,k_2,\ldots,k_i)}_{(p_1,p_2,\ldots,p_i)}$ that are not also i-boundaries always follow a copy of p_i. It follows that any $(i-1)$-boundary after a codeword $\tau_i \neq p_i$ must therefore also be an i-boundary. Thus if a decoder synchronised to level $i-1$ encounters such a codeword $\tau_i \neq p_i$, it transitions to state i.

Conversely, we observe that only p_i can prevent a transition from state $i-1$ to state i. To render a decoder permanently unsynchronised, the compressed message x must thus end in a run of p_i. Since semi-infinite runs of p_i are incompatible with any notion of a random semi-infinite sequence, a T-code decoder will transition from state $i-1$ to state i in such a sequence, and thus ultimately reach state n. T-codes are thus statistically self-synchronising. Furthermore, removing any periodic codewords from $A^{(k_1,k_2,\ldots,k_n)}_{(p_1,p_2,\ldots,p_n)}$ results in a BSD code from which one cannot construct a sequence with a semi-infinite run of of p_i. We will refer to such codes as BSD-T-codes below. However, note that each T-augmentation only creates one periodic codeword, whereas the total number of codewords nearly doubles at each step. Moreover, some of the periodic codewords created may in turn become T-prefixes in later T-augmentations and thus not be part of the final T-code. The removal of the remaining periodic codewords thus

generally only represents a minor change to the code without substantial effect on its compression potential.

An unsychronised T-code decoder will thus synchronise after any sequence of the form:

$$\sigma(r_1,\tau_1,r_2,\tau_2,\ldots,r_n,\tau_n) = p_1^{r_1}\tau_1 p_2^{r_2}\tau_2\ldots p_n^{r_n}\tau_n \quad (4)$$

where all r_i finite and $\tau_i \in A^{(k_1,k_2,\ldots,k_{i-1})}_{(p_1,p_2,\ldots,p_{i-1})}$ with $\tau_i \neq p_i$. If we remove all periodic codewords from $A^{(k_1,k_2,\ldots,k_n)}_{(p_1,p_2,\ldots,p_n)}$ (i.e., we convert the T-code into a BSD-T-code), any sufficiently long string over A starts with some $\sigma(r_1,\tau_1,r_2,\tau_2,\ldots,r_n,\tau_n)$, and there is a limit on the r_i that we may encounter.

Abbreviating our notation to $\sigma_n = \sigma(r_1,\tau_1,r_2,\tau_2,\ldots,r_n,\tau_n)$, the expected synchronisation delay of a T-code (with periodic codewords) is thus given by the expected length of σ_n, which is

$$E[|\sigma_n|] = \sum_{r_1,\tau_1,r_2,\tau_2,\ldots,r_n,\tau_n} P(\sigma_n)|\sigma_n|, \quad (5)$$

where $P(\sigma_n)$ is the probability that a particular σ_n occurs.

If the encoded T-code symbol stream is approximated as an i.i.d. Bernoulli source with equiprobable symbols, we may resolve this further as:

$$\begin{aligned} E[|\sigma_n|] &= \sum_{i=1}^{n} E[|p_i^{r_i}\tau_i|] = \sum_{i=1}^{n} E[|p_i^{r_i}|]E[|\tau_i|] \\ &= \sum_{i=1}^{n} E[|p_i^{r_i}|]E[|\tau_i|] = \sum_{i=1}^{n} |p_i|E[r_i]E[|\tau_i|]. \end{aligned} \quad (6)$$

As we have $0 \leq r_i < \infty$ and $P(p_i) = \#A^{-|p_i|}$, the probability $P(r_i = j)$ of r_i taking the value j is:

$$\begin{aligned} P(r_i = j) &= P(p_i^j)(1 - P(p_i)) = P(p_i)^j(1 - P(p_i)) \\ &= (1 - \#A^{-|p_i|})\#A^{-j|p_i|}. \end{aligned} \quad (7)$$

Thus

$$\begin{aligned} E[r_i] &= \sum_{j=0}^{\infty} jP(r_i = j) = \sum_{j=0}^{\infty} j(1 - \#A^{-|p_i|})\#A^{-j|p_i|} \\ &= (1 - \#A^{-|p_i|}) \sum_{j=1}^{\infty} j\#A^{-j|p_i|} \\ &= \frac{(1 - \#A^{-|p_i|})\#A^{-|p_i|}}{(1 - \#A^{-|p_i|})^2} = \frac{\#A^{-|p_i|}}{1 - \#A^{-|p_i|}} \end{aligned} \quad (8)$$

where we have used the well known polylogarithm series sum.

Furthermore, writing $P_i = \#A^{-|p_i|}$, we may then recursively derive the expected length of a codeword $\tau_i \neq p_i$ in $A^{(k_1,k_2,\ldots,k_{i-1})}_{(p_1,p_2,\ldots,p_{i-1})}$, $E[|\tau_i|]$ from $E[|\tau_{i-1}|]$ as follows:

$$\begin{aligned} E[|\tau_1|] &= 1 \\ E[|\tau_i|] &= \frac{(1 - P_{i-1}) \sum_{j=0}^{k_{i-1}} P_{i-1}^{-j}(E[|\tau_{i-1}|] + j|p_{i-1}|)}{1 - P_i} \\ &\quad + \frac{P_{i-1}^{-(k_{i-1}+1)}(k_{i-1}+1)|p_{i-1}| - P_i|p_i|}{1 - P_i}. \end{aligned} \quad (9)$$

In the case of BSD-T-codes, the $E[|\sigma_n|]$ derived for the general case represents an upper bound on the expected synchronisation delay. We can also put an upper bound on $|\sigma_n|$ itself in this case, by considering the maximum possible value for an r_i, i.e., the longest possible run of p_i. As we exclude periodic codewords, this run can only occur inside the overlap w_1w_2 of two non-periodic codewords $w_1, w_2 \in A^{(k_1,k_2,\ldots,k_n)}_{(p_1,p_2,\ldots,p_n)}$ and its length may thus be determined by inspection of $A^{(k_1,k_2,\ldots,k_n)}_{(p_1,p_2,\ldots,p_n)}$. As the boundary between w_1 and w_2 is an n- and hence an $(i-1)$-boundary, p_i cannot straddle it. The length of the potential run of p_i at the beginning of w_2 is limited by $\lfloor \frac{|w_2| - |w_{\min_i}|}{|p_i|} \rfloor$, where $w_{\min_i}$ is the shortest codeword other than p_i in $A^{(k_1,k_2,\ldots,k_{i-1})}_{(p_1,p_2,\ldots,p_{i-1})}$, i.e., the shortest word that could have been T-prefixed by p_i in the i-th T-augmentation. In w_1, the longest possible run is limited by $\lfloor \frac{|w_1| - |p_{p_i}|}{|p_i|} \rfloor$, where p_{p_i} is the smallest T-prefix used at any T-augmentation between the creation of p_i and level $i-1$. Given only the T-prefix lengths and the T-expansion parameters, we may still place a (loose) upper bound on $|\sigma_n|$ as $2n|w_{max}|$, where w_{max} is the length of the longest codewords in $A^{(k_1,k_2,\ldots,k_n)}_{(p_1,p_2,\ldots,p_n)}$.

Note that the synchronisation model presented here is a simplified one, as a decoder may transition from state i to state j immediately if it can exclude the possibility that τ_i might be the suffix of any of $p_{i+1}, p_{i+2}, \ldots p_j$ for some $j > i$. A detailed treatment of this extended model (see [18]) is beyond the proof-of-concept scope of this paper.

V. Adaptation of the matching algorithm

If we require a bounded synchronisation delay, we cannot use a full T-code but rather must remove any periodic codewords from the final code to obtain a BSD-T-code. Each T-augmentation creates one such codeword. Some of these may however become T-prefixes in later T-augmentations and may thus not be in the final code. However, their special status necessitates a number of changes to the matching algorithm and its associated feasibility criteria:

- No periodic codeword may be assigned to a source symbol, i.e., we must now assign any source symbol to a (potentially longer) different codeword if it would have been assigned to a periodic codeword in a complete T-code. As a consequence, we generally need a larger code. In particular, it necessitates a change in the fixed constraint on the maximal codeword length above to $k|p| + \ell_2 \leq \lceil (\#S + \phi - 1)/(\#A - 1) \rceil$, where ϕ is the number of periodic codewords in the final code before removal. As each T-augmentation creates one periodic codeword, the value of ϕ depends on the number of T-augmentations taken to arrive at the final set, which we cannot know in advance. Alternatively, we may consider the minimum total number of codewords required, and for each set compute the number of source symbols assigned to codewords shorter than the present T-prefix, $n_{<|p|}$. These assignments persist below the present node in the search tree as our T-prefixes are in non-decreasing order. Any remaining assignments must thus take place between

TABLE I
THE EFFICIENCY OF BSD-T-CODES AND HUFFMAN CODES

Source	32 symbols				
	$H(S)$	r_T[bits]	r_H[bits]	no. of δ_C	$\lvert\sigma_n\rvert$[bits]
Exponential $\frac{1}{2}$	2	$3.4{\cdot}10^{-10}$	$1{\cdot}10^{-10}$	1,225	$\leq$66
Exponential $\frac{2}{3}$	2.75	0.16	0.045	1,284	$\leq$46
Harmonic	4.15	0.14	0.019	1,304	$\leq$26
Linear	4.74	0.47	0.03	1,805	$\leq$26
Equiprobable	5	0.88	0	2,109	$\leq$26
	64 symbols				
	$H(S)$	r_T[bits]	r_H[bits]	no. of δ_C	$\lvert\sigma_n\rvert$[bits]
Harmonic	4.86	0.16	0.029	10,530	$\leq$54
Linear	5.73	0.57	0.029	11,934	$\leq$40
Equiprobable	6	0.98	0	16,891	$\leq$38
	128 symbols				
	$H(S)$	r_T[bits]	r_H[bits]	no. of δ_C	$\lvert\sigma_n\rvert$[bits]
Harmonic	5.55	0.16	0.045	91,337	$\leq$66
Linear	6.73	0.66	0.029	119,557	$\leq$46
Equiprobable	7	1.16	0	181,290	$\leq$62

the length of the current T-prefix $|p|$ and the maximum codeword length of interest. Since we can assign at least $\#A-1$ codewords of each length (with the possible exception of length $|p|$), the maximum length of interest becomes $\lceil |p|+n_{<|p|}/(\#A-1)\rceil$. Note that this constraint only applies to further VTA of small sets (see above).

- Every VTA needs to record the number of periodic codewords for each length. Further VTA must use these codewords as T-prefixes before any non-periodic codewords (otherwise, we would ultimately lose two codewords of this length rather than just one).

VI. EXAMPLES

How efficient are BSD-T-codes compared to Huffman codes, i.e., how much redundancy does guaranteed local decodability add? Using the algorithm above, we obtain the source entropy $H(S)$, the redundancies r_T and r_H for BSD-T-codes and Huffman codes, and the bound for $|\sigma_n|$ for a suitable BSD-T-code for the following i.i.d. source types for 32, 64, and 128 symbols:

- Two sources whose symbol probabilities tail off exponentially with a factor of $\frac{1}{2}$ and $\frac{2}{3}$, respectively (for 32 symbols only).
- A source whose symbol probabilities tail off harmonically (i'th largest probability is $\frac{1}{i}$ times the maximum probability).
- A source with linear tail-off (i'th smallest symbol probability is $i/\#S$).
- A source with equiprobable symbols.

These results are given in Table I along with the number of δ_C searched in each case. These results suggests that low entropy sources yield compression comparable to that of Huffman codes, whereas high entropy sources do not compress. These latter sources are however of less interest because of their limited general compressibility. Note that the matching algorithm returns only the optimal T-prefix lengths. This lets the code designer choose the actual T-prefixes to optimise for the best compromise between maximum and expected synchronisation delay.

VII. CONCLUSION

In this paper, we introduced variable-length locally decodable source codes and showed that T-codes are candidates for this purpose. Overall, the trade-off in efficiency to provide local decodability seems modest. Our results also show that the search effort of the matching algorithm is lowest for the most compressible sources, but increases rapidly with source cardinality. While the search algorithm needs to search a large number of sets to confirm the optimal BSD-T-code for a given probability distribution in terms of redundancy, in practice it discovers codes with low redundancies rather quickly. For example, a harmonic source with 256 symbols and an entropy of 6.22 bits/symbol can be encoded in this way with a redundancy of less than a sixth of a bit. Also the search algorithm is suited to parallelisation, an avenue we have not yet investigated.

REFERENCES

[1] A. Makhdoumi, S. L. Huang, M. Médard, and Y. Polyanskiy, "On locally decodable source coding," *arXiv:1308.5239 [cs.IT]*, 2013.

[2] Y. Kaspi and N. Merhav, "Zero-delay and causal single-user and multi-user lossy source coding with decoder side information," *arXiv:1301.0079 [cs.IT]*, 2013.

[3] D. L. Neuhoff and R. Gilbert, "Causal source codes," *IEEE Trans. Inform. Theory*, vol. 28, no. 5, pp. 701–713, Sep. 1982.

[4] A. Mazumdar, G. W. Wornell, and V. Chandar, "Update efficient codes for error correction," in *Proc. Int. Symp. on Inform. Theory*, Cambridge, MA, pp. 1558–1662, Jul. 2012

[5] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, Jul. 1970.

[6] R. Pagh, "Low redundancy in static dictionaries with constant query time," *SIAM J. Comput.*, vol. 31, no. 2, pp. 353–363, 2001.

[7] M. Patrascu, "Succincter," in *Proc. IEEE Symp. on Foundations of Computer Science*, Philadelphia, PA, pp. 305–313, Oct. 2008.

[8] V. Chandar, D. Shah, and G. W. Wornell, " A locally encodable and decodable compressed data structure," in *Proc. Allerton Conf. on Commun., Control, and Comput.*, Monticello, IL, pp. 613–619, Sep.-Oct. 2009.

[9] D. Huffman, "A method for the construction of minimum redundancy codes," *Proc. IRE*, vol. 40, no. 9, pp. 1098–1101, Sep. 1952.

[10] U. Speidel, "A note on BSD codes constructed from T-codes," in *Proc. IEEE Int. Symp. on Inform. Theory*, Seoul, Korea, pp. 2418–2421, Jun.-Jul. 2009.

[11] M. R. Titchener, "Digital encoding by means of new T-codes to provide improved data synchronisation and message integrity", *IEE Proc. – Computers and Digital Tech.*, vol. 131, no. 4, pp. 151–153, Jul. 1984.

[12] R. Nicolescu, Uniqueness Theorems for T-codes. Technical Report. Tamaki Report Series no. 9, The University of Auckland, 1995.

[13] M. R. Titchener, "Generalized T-Codes: An extended construction algorithm for self-synchronizing variable-length codes," *IEE Proc. – Computers and Digital Tech.*, vol. 143, no. 3, pp. 122–128, Jun. 1998.

[14] U. Guenther, "Data compression and serial communication with generalized T-Codes," *J. Universal Computer Science*, vol. 2, no. 11, pp. 769–795, 1996.

[15] G. R. Higgie, "Database of best T-codes," *IEE Proc. – Computers and Digital Tech.*, vol. 143, no. 4, pp. 213–218, Jul. 1996.

[16] M. R. Titchener, *verbal communication*.

[17] R. Nicolescu and M. R. Titchener, "Uniqueness theorems for T-codes," *Romanian J. Inform. Science and Tech.*, vol. 1, no. 3, pp. 243–258, Mar. 1998.

[18] U. Guenther, Robust Source Coding with Generalized T-codes. PhD Thesis, The University of Auckland, 1998.

[19] U. Guenther, "Matching T-codes to a source," in *Proc. Int. Conf. on Inform., Commun., and Signal Process.*, Singapore, paper P0156, Oct. 2001.