Large Alphabet Source Coding Using Independent Component Analysis

Amichai Painsky, *Member, IEEE*, Saharon Rosset, and Meir Feder, *Fellow, IEEE*

***Abstract*— Large alphabet source coding is a basic and well-studied problem in data compression. It has many applications, such as compression of natural language text, speech, and images. The classic perception of most commonly used methods is that a source is best described over an alphabet, which is at least as large as the observed alphabet. In this paper, we challenge this approach and introduce a conceptual framework in which a large alphabet source is decomposed into "as statistically independent as possible" components. This decomposition allows us to apply entropy encoding to each component separately, while benefiting from their reduced alphabet size. We show that in many cases, such decomposition results in a sum of marginal entropies which is only slightly greater than the entropy of the source. Our suggested algorithm, based on a generalization of the binary independent component analysis, is applicable for a variety of large alphabet source coding setups. This includes the classical lossless compression, universal compression, and high-dimensional vector quantization. In each of these setups, our suggested approach outperforms most commonly used methods. Moreover, our proposed framework is significantly easier to implement in most of these cases.**

***Index Terms*— Data Compression, Source Coding, Entropy Coding, Independent Component Analysis.**

I. Introduction

ASSUME a source over an alphabet size m, from which a sequence of n independent samples are drawn. The classical source coding problem is concerned with finding a sample-to-codeword mapping, such that the average codeword length is minimal, and the samples may be uniquely decodable. This problem was studied since the early days of information theory, and a variety of algorithms [1], [2] and theoretical bounds [3] were introduced throughout the years.

The classical source coding problem usually assumes an alphabet size m which is small, compared with n. Here, we focus on a more difficult (and common) scenario, where the source's alphabet size is considered "large" (for example, a word-wise compression of natural language texts). In this setup, m takes values which are either comparable, or even larger, than the length of the sequence n. The main challenge in large alphabet source coding is that the redundancy of the code, formally defined as the excess number of bits used over the source's entropy, typically increases with the alphabet size [4].

In this work we propose a conceptual framework for large alphabet source coding, in which we reduce the alphabet size by decomposing the source into multiple components which are "as statistically independent as possible". This allows us to encode each of the components separately, while benefiting from the reduced redundancy of the smaller alphabet of each component.

To utilize this concept we introduce a framework based on a generalization of the Binary Independent Component Analysis (BICA) method [5]. This framework efficiently searches for an invertible transformation which minimizes the difference between the sum of marginal entropies (after the transformation is applied) and the joint entropy of the source. Hence, it minimizes the (attainable) lower bound on the average codeword length, when applying marginal entropy coding.

We show that while there exist sources which cannot be efficiently decomposed, their portion (of all possible sources over a given alphabet size) is small. Moreover, we show that the difference between the sum of marginal entropies (after our transformation is applied) and the joint entropy, is bounded, on average, by a small constant for every m (when the averaging takes place uniformly, over all possible sources of the same alphabet size m). This implies that our suggested approach is suitable for many sources of increasing alphabet size. Our analysis is based on the *order permutation*, which is known to be the optimal solution of the Guessing problem [6]. In this work we provide an additional important statistic (the sum of marginal entropies) for the Guessing problem.

We demonstrate our method in a variety of large alphabet source coding setups. This includes the classical lossless coding, when the probability distribution of the source is known both to the encoder and the decoder, universal lossless coding, in which the decoder is not familiar with the distribution of the source, and lossy coding in the form of vector quantization. We show that our approach outperforms currently known methods in all these setups, for a variety of typical sources.

The rest of this manuscript is organized as follows: After a short notation section, we review the work that was previously done in large alphabet source coding in Section III. Section IV presents the generalized BICA problem, proposes two different solutions, and demonstrates their behavior on average and in the worst-case. In Section V we apply our suggested

Manuscript received September 5, 2015; revised November 23, 2016 and July 5, 2017; accepted July 11, 2017. Date of publication July 18, 2017; date of current version September 13, 2017. A. Painsky was supported in part by the Israeli Ministry of Science and in part by the Israeli Science Foundation under Grant 1487/12. This paper was presented in part at the 2014 International Symposium on Information Theory and the 2015 Data Compression Conference.

A. Painsky was with the Statistics Department, Tel Aviv University, Tel Aviv 6997801, Israel. He is now with the Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge, MA 02139, USA (e-mail: amichai@mit.edu).

S. Rosset is with the Statistics Department, Tel Aviv University, Tel Aviv 6997801, Israel.

M. Feder is with the Department of Electrical Engineering, Tel Aviv University, Tel Aviv 6997801, Israel.

Communicated by E. Tuncel, Associate Editor for Source Coding.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIT.2017.2728017

framework to the classical lossless coding problem, over large alphabets. We then extend the discussion to universal compression in Section VI. Finally, Section VII demonstrates our approach on vector quantization, with a special attention to high dimensional sources and low distortion.

II. Notation

Throughout this work we use the following standard notation: underlines denote vector quantities, where their respective components are written without underlines but with index. For example, the components of the d-dimensional vector $\underline{X}$ are $X_1, X_2, \dots X_d$. Random variables are denoted with capital letters while their realizations are denoted with the respective lower-case letters. $P_{\underline{X}}\left(\underline{x}\right) \triangleq P(X_1 = x_1, X_2 = x_2 \dots)$ is the probability function of $\underline{X}$ while $H\left(\underline{X}\right)$ is the entropy of $\underline{X}$. This means $H\left(\underline{X}\right) = -\sum_{\underline{x}} P_{\underline{X}}\left(\underline{x}\right) \log P_{\underline{X}}\left(\underline{x}\right)$ where the log function denotes a logarithm of base 2 and $\lim_{x \to 0} x \log(x) = 0$. Specifically, we refer to the binary entropy of a Bernoulli distributed random variable $X \sim \text{Ber}(p)$ as $H_b(X)$, while we denote the binary entropy function as $h_b\ (p) = -p \log p - (1-p)\log(1-p)$.

III. Previous Work

In the classical lossless data compression framework, one usually assumes that both the encoder and the decoder are familiar with the probability distribution of the encoded source, $\underline{X}$. Therefore, encoding a sequence of n memoryless samples drawn form this this source takes on average at least n times its entropy $H\left(\underline{X}\right)$, for sufficiently large n [3]. In other words, if n is large enough to assume that the joint empirical entropy of the samples, $\hat{H}\left(\underline{X}\right)$, is close enough to the true joint entropy of the source, $H\left(\underline{X}\right)$, then $H\left(\underline{X}\right)$ is the minimal average number of bits required to encode a source symbol. Moreover, it can be shown [3] that the minimum average codeword length, $\bar{l}_{min}$, for a uniquely decodable code, satisfies

$$H\left(\underline{X}\right) \leq \bar{l}_{min} \leq H\left(\underline{X}\right) + 1. \tag{1}$$

Entropy coding is a lossless data compression scheme that strives to achieve the lower bound, $\bar{l}_{min} = H\left(\underline{X}\right)$. Two of the most common entropy coding techniques are Huffman coding [1] and arithmetic coding [2]. The Huffman algorithm is an iterative construction of variable-length code table for encoding the source symbols. The algorithm derives this table from the probability of occurrence of each source symbol. Assuming these probabilities are dyadic (i.e., $-\log\ p(\underline{x})$ is an integer for every symbol $\underline{x} \in \underline{X}$), then the Huffman algorithm achieves $\bar{l}_{min} = H\left(\underline{X}\right)$. However, in the case where the probabilities are not dyadic then the Huffman code does not achieve the lower-bound of (1) and may result in an average codeword length of up to $H\left(\underline{X}\right) + 1$ bits. Moreover, although the Huffman code is theoretically easy to construct (linear in the number of symbols, assuming they are sorted according to their probabilities) it is practically a challenge to implement when the number of symbols increases [7]. Huffman codes achieve the minimum average codeword length among all uniquely decodable codes that assign a separate codeword to each symbol. However, if the probability of one of the symbols is close to 1, a Huffman code with an average codeword length close to the entropy can only be constructed if a large number of symbols is jointly coded. The popular method of arithmetic coding is designed to overcome this problem.

In arithmetic coding, instead of using a sequence of bits to represent a symbol, we represent it by a subinterval of the unit interval [2]. This means that the code for a sequence of symbols is an interval whose length decreases as we add more symbols to the sequence. This property allows us to have a coding scheme that is incremental. In other words, the code for an extension to a sequence can be calculated simply from the code for the original sequence. Moreover, the codeword lengths are not restricted to be integral. The arithmetic coding procedure achieves an average length for the block that is within 2 bits of the entropy. Although this is not necessarily optimal for any fixed block length (as we show for Huffman code), the procedure is incremental and can be used for any block-length. Moreover, it does not require the source probabilities to be dyadic. However, arithmetic codes are more complicated to implement and are a less likely to practically achieve the entropy of the source as the number of symbols increases. More specifically, due to the well-known underflow and overflow problems, finite precision implementations of the traditional adaptive arithmetic coding cannot work if the size of the source exceeds a certain limit [8]. For example, the widely used arithmetic coder by Witten *et al.* [2] cannot work when the alphabet size is greater than 2^{15}. The improved version of arithmetic coder by Moffat *et al.* [9] extends the alphabet to size 2^{30} by using low precision arithmetic, at the expense of compression performance.

Notice that a large number of symbols not only results in difficulties in implementing entropy codes: as the alphabet size increases, we require a growing number of samples for the empirical entropy to converge to the true entropy. Therefore, when dealing with sources over large alphabets we usually turn to a universal compression framework. Here, we assume that the empirical probability distribution is not necessarily equal to the true distribution and henceforth unknown to the decoder. This means that a compressed representation of the samples now involves with two parts – the compressed samples and an overhead redundancy (where the redundancy is defined as difference between the number of bits used to transmit a message and the entropy of the sequence).

As mentioned above, encoding a sequence of n samples, drawn from a memoryless source $\underline{X}$, requires at least n times the empirical entropy, $\hat{H}(\underline{X})$. Assuming that an optimal codebook is assigned for the sequence, after it is known, $n\hat{H}(\underline{X})$ is also the minimal code length of the sequence. The redundancy, on the other hand, may be analyzed in several ways. One common way is through the minimax criterion [4]. Here, the *worst-case redundancy* is the lowest number of extra bits (over the empirical entropy) required in the worst case (that is, among all sequences) by any possible encoder. Many worst-case redundancy results are known when the source's alphabet is finite. A succession of papers initiated by Shtarkov [10] show that for the collection $\mathcal{I}_m^n$ of i.i.d. distributions over length-n sequences drawn from an alphabet of a fixed size m, the worst-case redundancy behaves asymptotically as $\frac{m-1}{2} \log \frac{n}{m}$, as n grows. Orlitsky and Santhanam [11] extended this result to cases where m varies with n. The *standard compression* scheme they introduce differentiates between three situations in which $m = o(n)$, $n = o(m)$ and $m = \Theta(n)$. They provide leading term asymptotics and bounds to the worst-case minimax redundancy for these ranges of the alphabet size. Szpankowski and Weinberger [12] completed this study, providing the precise asymptotics to these ranges. For the purpose of our work we adopt the leading terms of their results, showing that the worst-case minimax redundancy, when $m \to \infty$, as n grows, behaves as follows: i

1) For $m = o(n)$:

$$\hat{R}(\mathcal{I}_m^n) \backsimeq \frac{m-1}{2}\log\frac{n}{m} + \frac{m}{2}\log e + \frac{m\log e}{3}\sqrt{\frac{m}{n}} \qquad (2)$$

2) For $n = o(m)$:

$$\hat{R}(\mathcal{I}_m^n) \backsimeq n\log\frac{m}{n} + \frac{3}{2}\frac{n^2}{m}\log e - \frac{3}{2}\frac{n}{m}\log e \qquad (3)$$

3) $m = \alpha n + l(n)$:

$$\hat{R}(\mathcal{I}_m^n) \backsimeq n\log B_\alpha + l(n)\log C_\alpha - \log\sqrt{A_\alpha} \qquad (4)$$

where α is a positive constant, $l(n) = o(n)$ and

$$C_\alpha \triangleq \frac{1}{2} + \frac{1}{2}\sqrt{1 + \frac{4}{\alpha}}, \quad A_\alpha \triangleq C_\alpha + \frac{2}{\alpha}, \quad B_\alpha \triangleq \alpha C_\alpha^{\alpha+2} e^{-\frac{1}{C_\alpha}}.$$

In addition to these theoretical minimax bounds, there exist several practical methods for universal compression. In his paper from 2004, [13] presented a novel framework for universal compression of memoryless sources over unknown and possibly infinite alphabets. According to their framework, the description of any string, over any alphabet, can be viewed as consisting of two parts: the symbols appearing in the string and the pattern that they form. For example, the string "abracadabra" can be described by conveying the *pattern* "12314151231" and the *dictionary*

index	1	2	3	4	5
letter	a	b	r	c	d

Together, the pattern and dictionary specify that the string "abracadabra" consists of the first letter to appear (a), followed by the second letter to appear (b), then by the third to appear (r), the first that appeared (a again), the fourth (c), etc. Therefore, a compressed string involves with a compression of the pattern and its corresponding dictionary. Orlitsky et al. derived the bounds for pattern compression, showing that the redundancy of patterns compression under i.i.d. distributions over potentially infinite alphabets is bounded by $\left(\frac{3}{2}\log e\right) n^{1/3}$. Therefore, assuming the alphabet size is m and the number of uniquely observed symbols is n_0, the dictionary can be described in $n_0 \log m$ bits, leading to an overall lower bound of $n_0 \log m + n^{1/3}$ bits on the compression redundancy.

An additional (and very common) universal compression scheme is the canonical Huffman coding [14]. A canonical Huffman code is a variant of Huffman code with unique properties which allow it to be described in a very compact manner. The advantage of a canonical Huffman tree is that one can encode a codebook in fewer bits than a fully described tree. Since a canonical Huffman codebook can be stored especially efficiently, most compressors start by generating a non-canonical Huffman codebook, and then convert it to a canonical form before using it. In canonical Huffman coding the bit lengths of each symbol are the same as in the traditional Huffman code. However, each code word is replaced with new code words (of the same length), such that a subsequent symbol is assigned the next binary number in sequence. For example, assume a Huffman code for four symbols, A to D:

symbol	A	B	C	D
codeword	11	0	101	100

Applying canonical Huffman coding to it we get

symbol	B	A	C	D
codeword	0	10	110	111

This way we do not need to store the entire Huffman mapping but only a list of all symbols in increasing order by their bit-lengths and record the number of symbols for each bit-length. This allows a more compact representation of the code, hence, lower redundancy.

An additional class of data encoding problems which we refer to in this work is lossy compression. In the lossy compression setup one applies inexact approximations for representing the content that has been encoded. In this work we focus on vector quantization, in which a high-dimensional vector $\underline{X} \in \mathbb{R}^d$ is to be represented by a finite number of points. Vector quantization works by clustering the observed samples of the vector $\underline{X}$ into groups, where each group is represented by its centroid point, such as in k-means and other clustering algorithms. Then, the centroid points that represent the observed samples are compressed in a lossless manner. In the lossy compression setup, one is usually interested in minimizing the amount of bits which represent the data for a given a distortion measure (or equivalently, minimizing the distortion for a given compressed data size). The rate-distortion function defines the lower bound on this objective. It is defined as

$$R(D) = \min_{P(\underline{Y}|\underline{X})} I(\underline{X};\underline{Y}) \quad s.t.\ \mathbb{E}\left\{D(\underline{X},\underline{Y})\right\} \leq D \qquad (5)$$

where $\underline{X}$ is the source, $\underline{Y}$ is recovered version of $\underline{X}$ and $D(\underline{X},\underline{Y})$ is some distortion measure between $\underline{X}$ and $\underline{Y}$. Notice that since the quantization is a deterministic mapping between $\underline{X}$ and $\underline{Y}$, we have that $I(\underline{X};\underline{Y}) = H(\underline{Y})$.

The Entropy Constrained Vector Quantization (ECVQ) is an iterative method for clustering the observed samples from $\underline{X}$ into centroid points which are later represented by a minimal average codeword length. The ECVQ algorithm minimizes the Lagrangian

$$L = \mathbb{E}\left\{D(\underline{X},\underline{Y})\right\} + \lambda\mathbb{E}\left\{l(\underline{X})\right\} \qquad (6)$$

where λ is the Lagrange multiplier and $\mathbb{E}\left(l(\underline{X})\right)$ is the average codeword length for each symbol in $\underline{X}$. The ECVQ algorithm performs an iterative local minimization method similar to the generalized Lloyd algorithm [15]. This means that for a given clustering of samples it constructs an entropy code to minimize the average codeword lengths of the centroids. Then, for a given coding of centroids it clusters the observed samples such that the average distortion is minimized, biased by the length of the codeword. This process continues until a local convergence occurs.

The ECVQ algorithm performs local optimization (as a variant of the k-means algorithm) which is also not very scalable for an increasing number of samples. This means that in the presence of a large number of samples, or when the alphabet size of the samples is large enough, the clustering phase of the ECVQ becomes impractical. Therefore, in these cases, one usually uses a predefined lattice quantizer and only constructs a corresponding codebook for its centroids.

It is quite evident that large alphabet sources entails a variety of difficulties in all the compression problems mentioned above: it is more complicated to construct an entropy code for, it results in a great redundancy when universally encoded and it is much more challenging to design a vector quantizer for. In the following sections we introduce a framework which is intended to overcome these drawbacks.

IV. Generalized Binary Independent Component Analysis

A common implicit assumption to most compression schemes in that the source is best represented over its observed alphabet size. We would like to challenge this assumption, suggesting that in some cases there exists a transformation which decomposes a source into multiple "as independent as possible" components whose alphabet size is much smaller.

A. Problem Formulation

Suppose we are given a binary random vector $\underline{X} \sim \underline{p}$ of a dimension d. We are interested in an invertible transformation $\underline{Y} = g(\underline{X})$ such that $\underline{\mathrm{Y}}$ is of the same dimension and alphabet size, $g : 2^d \to 2^d$. In addition we would like the components (bits) of $\underline{\mathrm{Y}}$ to be as "statistically independent as possible". Notice that an invertible transformation of a vector $\underline{\mathrm{X}}$ is actually a one-to-one mapping (i.e. permutation) of its 2^d alphabet symbols. Therefore, there exist $2^d!$ possible invertible transformations.

To quantify the statistical independence among the components of the vector $\underline{\mathrm{Y}}$ we use the well-known *total correlation* measure as a multivariate generalization of the mutual information,

$$C(\underline{Y}) = \sum_{j=1}^{d} H_b(Y_j) - H(\underline{Y}). \tag{7}$$

This measure can also be viewed as the cost of encoding the vector $\underline{Y}$ component-wise, as if its components were statistically independent, compared to its true entropy. Notice that the total correlation is non-negative and equals zero iff the components of $\underline{Y}$ are mutually independent. Therefore, "as statistically independent as possible" may be quantified by minimizing $C(\underline{Y})$. The total correlation measure was first considered as an objective for minimal redundancy representation by Barlow [16]. It is also equivalent to the Kullback-Leibler divergence between joint distribution and product of its marginals [17].

Since we define $\underline{\mathrm{Y}}$ to be an invertible transformation of $\underline{\mathrm{X}}$ we have $H(\underline{Y}) = H(\underline{X})$ and our minimization objective is

$$\sum_{j=1}^{d} H_b(Y_j) \to min. \tag{8}$$

We notice that $P(Y_j = 0)$ is the sum of probabilities of all words whose j^{th} bit equals 0. We further notice that the optimal transformation is not unique. For example, we can always invert the j^{th} bit of all words, or even shuffle the bits, to achieve the same minimum.

In the following sections we review and introduce several methods for solving (8). As a first step towards this goal we briefly review the generalized BICA method. A complete derivation of this framework appears in [18]. Then, Sections IV-C,IV-D and IV-E provide a simplified novel method for minimizing (8) and discuss its theoretical properties.

B. Piece-Wise Linear Relaxation Algorithm

In this section we review our previously suggested method for minimizing (8), as it appears in detail in [18]. Let us first notice that (8) is a concave minimization problem over a discrete permutation set which is a hard problem. However, let us assume for the moment that instead of minimizing our "true" objective, we have a simpler linear objective function such as

$$L(\underline{Y}) = \sum_{j=1}^{d} a_j \pi_j + b_j = \sum_{i=1}^{2^d} c_i P(\underline{Y} = y(i)) + d \tag{9}$$

where $\pi_j \equiv p(Y_j = 0)$ and the last equality changes the summation over d bits to a summation over all 2^d symbols. In order to minimize this objective function over the 2^d given probabilities $\underline{p}$, we simply sort the values of $\underline{p}$ in a descending order and allocate them such that the largest probability goes with the smallest coefficient c_i and so on. Assuming both the coefficients c_i and the probabilities p_i are known and sorted in advance, the complexity of this procedure is linear in 2^d.

We now turn to the generalized BICA problem, defined in (8). Since our objective is concave we would first like to bound it from above with a piecewise linear function that contains k pieces, as shown in Figure 1. We show that solving the piecewise linear problem approximates the solution to (8) as closely as we want, depending on the number of pieces k.

First, notice that all $\pi_j's$ are exchangeable (in the sense that we can always interchange them and achieve the same result). This means we can find the optimal solution to the piece-wise linear problem by going over all possible combinations of "placing" the d variables π_j in the k different regions of the piece-wise linear function. For each of these combinations we need to solve a linear problem (9), where the minimization is with respect to allocation of the m given probabilities $\underline{p}$, and with additional constraints on the ranges of each π_j. For example, assume $d = 3$ and the optimal solution is such that

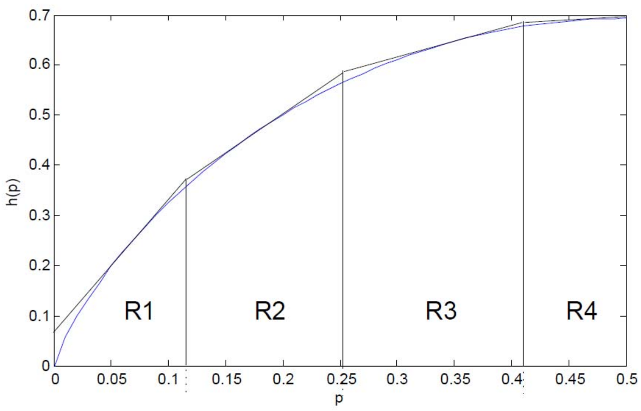

Fig. 1. piecewise linear ($k = 4$) relaxation to the binary entropy.

two $\pi_j's$ (e.g. π_1 and π_2) are at the first region R_1 and π_3 is at the second region R_2 , then we need to solve the following constrained linear problem,

$$\begin{aligned}&\text{minimize}a_1 \cdot (\pi_1 + \pi_2) + 2b_1 + a_2 \cdot \pi_3 + b_2\\&\text{subject to}\pi_1, \pi_2 \in R_1, \pi_3 \in R_2\end{aligned} \tag{10}$$

where the minimization is over the allocation of the given $\{p_i\}_{i=1}^{2^d}$, which determine the corresponding π_j's, as demonstrated in (9). This problem again seems hard. However, if we attempt to solve it without the constraints we notice the following:

1) If the collection of $\pi_j's$ which define the optimal solution to the unconstrained linear problem happens to meet the constraints then it is obviously the optimal solution with the constraints.
2) If the collection of $\pi_j's$ of the optimal solution does not meet the constraints (say, $\pi_2 \in R_2$) then, due to the concavity of the entropy function, there exists a different combination with a different constrained linear problem,

$$\begin{aligned}&\text{minimize}a_1\pi_1 + b_1 + a_2(\pi_2 + \pi_3) + 2b_2\\&\text{subject to}\pi_1 \in R_1\ \pi_2, \pi_3 \in R_2\end{aligned}$$

in which this set of $\pi_j's$ necessarily achieves a lower minimum (since $a_2\ x + b_2 < a_1\ x + b_1\ \forall x \in R_2$).

Therefore, in order to find the optimal solution to the piecewise linear problem, all we need to do is to go over all possible combinations of placing the $\pi_j's$ in k different regions, and for each combination solve an unconstrained linear problem (which is solved in a linear time in 2^d). If the solution does not meet the constraint then it means that the assumption that the optimal π_j's reside within this combination's regions is false. Otherwise, if the solution does meet the constraint, it is considered as a candidate for the global optimal solution.

The number of combinations we need to go through is equivalent to the number of ways of placing d identical balls in k boxes, which is (for a fixed k),

$$\binom{d+k-1}{d} = O(d^k). \tag{11}$$

Assuming the coefficients are all known and sorted in advance, the overall complexity of our suggested algorithm, as d increases, is just $O(d^k \cdot 2^d)$.

Notice that any approach that exploits the full statistical description of $\underline{X}$ would require going over the probabilities of all of its symbols at least once. Therefore, a computational load of at least $O(2^d)$ seems inevitable. Still, this is significantly smaller then $O(2^d!)$, required by brute-force search over all possible permutations.

It is also important to notice that even though the asymptotic complexity of our approximation algorithm is $O(d^k \cdot 2^d)$, it only takes a few seconds to run an entire experiment on a standard personal computer (with $d = 1024$ and $k = 8$, for example). The reason is that the 2^d term comes from the complexity of sorting a vector and multiplying two vectors, operations which are computationally efficient on most available software. Moreover, if we assume that the linear problems coefficients (9) are calculated, sorted and stored in advance, we can place them in a matrix form and multiply the matrix with the (sorted) vector $\underline{p}$. The minimum of this product is exactly the solution to the linear approximation problem. Therefore, the practical asymptotic complexity of the approximation algorithm drops to a single multiplication of a $(d^k \times 2^d)$ matrix with a $(2^d \times 1)$ vector. Even though the complexity of this method is significantly lower than full enumeration, it may still be computationally infeasible as d increases. Therefore, we suggest a simpler (greedy) solution, which is much easier to implement and apply.

C. Order Algorithm

As mentioned above, the minimization problem we are dealing with (8) is combinatorial in its essence and is consequently considered hard. We therefore suggest a simplified greedy algorithm which strives to sequentially minimize each term of the summation (8), $H_b(Y_j)$, for $j = 1, \dots, d$.

With no loss of generality, let us start by minimizing $H_b(Y_1)$, which corresponds to the marginal entropy of the most significant bit (msb). Since the binary entropy is monotonically increasing in the range $\left[0, \frac{1}{2}\right]$, we would like to find a permutation of $\underline{p}$ that minimizes the sum of half of its values. This means we should order the p_i's so that half of the p_i's with the smallest values are assigned to $P(Y_1) = 0$ while the other half of p_i's (with the largest values) are assigned to $P(Y_1) = 1$. For example, assuming $d = 3$ and $p_1 \le p_2 \le \cdots \le p_8$, a permutation which minimizes $H_b(Y_1)$ is

codeword	000	001	010	011	100	101	110	111
probability	p_2	p_3	p_1	p_4	p_8	p_5	p_6	p_7

We now proceed to minimize the marginal entropy of the second most significant bit, $H_b(Y_2)$. Again, we would like to assign $P(Y_2) = 0$ the smallest possible values of p_i's. However, since the we already determined which p_i's are assigned to the msb, all we can do is reorder the p_i's without changing the msb. This means we again sort the p_i's so that the smallest possible values are assigned to $P(Y_2) = 0$, without changing the msb. In our example, this leads to,

codeword	000	001	010	011	100	101	110	111
probability	p_2	p_1	p_3	p_4	p_6	p_5	p_8	p_7

Continuing in the same manner, we would now like to reorder the p_i's to minimize $H_b(Y_3)$ without changing the previous bits. This results in

codeword	000	001	010	011	100	101	110	111
probability	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8

Therefore, we show that a greedy solution to (8) which sequentially minimizes $H_b(Y_j)$ is attained by simply ordering the joint distribution $\underline{p}$ in an ascending (or equivalently descending) order. In other words, the *order permutation* suggests to simply order the probability distribution $p_1, \dots, p_{2^d}$ in an ascending order, followed by a mapping of the i^{th} symbol (in its binary representation) the i^{th} smallest probability.

The order permutation is not new to the Information Theory community. In fact, one of its key applications is due to the Guessing problem. Assume a password is chosen at random from a finite word-set of 2^d words. Then, the optimal strategy for guessing the chosen password is simply to guess passwords in decreasing order of probability. The number of attempts required to guess the correct password is called the *Guesswork* of the word-set. Several statistics of the Guesswork have been studied over the years. Massey [6] proved that the Shannon entropy of the word-set bounds from below the expected Guesswork, and that no general upper bound exists. Later, Arikan [19] extended the analysis to an asymptotic regime in which sequence of passwords is chosen at random with i.i.d. letters. He showed that in this setup, the moments of the Guesswork are closely related to the Rényi entropy of a single letter. This result was subsequently extended by Malone and Sullivan [20], Pfister and Sullivan [21], and Christiansen and Duffy [22]. Besides being an optimal solution to the Guessing problem, we notice that the Guesswork can be viewed as an implementation of our suggested ordering method. Just like the Guesswork, our suggested approach orders the probabilities. Then, we match each probability value to a symbol indicating its location in the order – just like Guesswork does in the context of the required number of guesses. In the following sections we provide a detailed analysis of several theoretical properties of our ordering method. These properties can also be regarded as an analysis of an additional statistic (the sum of marginal entropies) of the Guesswork of a word-set.

D. Worst-Case Independent Representation

We now introduce the theoretical properties of our suggested algorithm. Naturally, we would like to quantify how much we "lose" by representing a given random vector $\underline{X}$ as if its components are statistically independent. Therefore, for any given random vector $\underline{X} \sim \underline{p}$ and an invertible transformation $\underline{Y} = g(\underline{X})$, we denote the cost function $C(\underline{p}, g) = \sum_{j=1}^{d} H_b(Y_j) - H(\underline{X})$, as appears in (8). For simplicity of presentation we use the notation $m \equiv 2^d$ to directly refer to the alphabet size.

Since both our methods strongly depend on the given probability distribution $\underline{p}$, we focus on the worst-case and the average case of $C(\underline{p}, g)$, with respect to $\underline{p}$. Let us denote the order permutation as g_{ord} and the permutation found by the piece-wise linear relaxation as g_{lin}. We further define g_{bst} as the permutation that results with a lower value of $C(\underline{p}, g)$, between g_{lin} and g_{ord}. This means that

$$g_{bst} = \underset{\{g_{lin}, g_{ord}\}}{\arg\min}\, C(\underline{p}, g).$$

In addition, we define g_{opt} as the optimal permutation that minimizes (8) over all possible permutations. Therefore, for any given $\tilde{\underline{p}}$, we have that $C(\tilde{\underline{p}}, g_{opt}) \leq C(\tilde{\underline{p}}, g_{bst}) \leq C(\tilde{\underline{p}}, g_{ord})$. In this Section we examine the worst-case performance of both of our suggested algorithms. Specifically, we would like to quantify the maximum of $C(\underline{p}, g)$ over all joint probability distributions $\underline{p}$, of a given alphabet size m.

Theorem 1: For any random vector $\underline{X} \sim \underline{p}$, over an alphabet size m we have that

$$\max_{\underline{p}} C(\underline{p}, g_{opt}) = \Theta(\log(m))$$

Proof: We first notice that $\sum_{j=1}^{d} H_b(Y_j) \leq d = \log(m)$. In addition, $H(\underline{X}) \geq 0$. Therefore, we have that $C(\underline{p}, g_{opt})$ is bounded from above by $\log(m)$. Let us also show that this bound is tight, in the sense that there exists a joint probability distribution $\tilde{\underline{p}}$ such that $C(\tilde{\underline{p}}, g_{opt})$ is linear in $\log(m)$. Let $\tilde{p}_1 = \tilde{p}_2 = \cdots = \tilde{p}_{m-1} = \frac{1}{3(m-1)}$ and $\tilde{p}_m = \frac{2}{3}$. Then, $\tilde{\underline{p}}$ is ordered and satisfies $P(Y_i = 0) = \frac{m}{6(m-1)}$.

In addition, we notice that assigning symbols in a decreasing order to $\tilde{\underline{p}}$ (as mentioned in Section IV-C) results with an optimal permutation. This is simply since $P(Y_j = 0) = \frac{m}{6(m-1)}$ is the minimal possible value of any $P(Y_j = 0)$ that can be achieved when summing any $\frac{m}{2}$ elements of $\tilde{p}_i$. Further we have that,

$$\begin{aligned} &C(\tilde{\underline{p}}, g_{opt}) \\ &\quad = \sum_{j=1}^{d} H_b(Y_j) - H(\underline{X}) = \log(m) \cdot h_b\left(\frac{m}{6(m-1)}\right) \\ &\qquad + \left((m-1)\frac{1}{3(m-1)} \log \frac{1}{3(m-1)} + \frac{2}{3} \log \frac{2}{3}\right) \\ &\quad = \log(m) \cdot h_b\left(\frac{m}{6(m-1)}\right) - \frac{1}{3}\log(m-1) \\ &\qquad + \frac{1}{3} \log \frac{1}{3} + \frac{2}{3} \log \frac{2}{3} \underset{m\to\infty}{\longrightarrow} \\ &\qquad \log(m) \cdot \left(h_b\left(\frac{1}{6}\right) - \frac{1}{3}\right) - h_b\left(\frac{1}{3}\right). \end{aligned} \tag{12}$$

Therefore, $\max_{\underline{p}} C(\underline{p}, g_{opt}) = \Theta(\log(m))$. ■

Theorem 1 shows that even the optimal permutation achieves a sum of marginal entropies which is $\Theta(\log(m))$ bits greater than the joint entropy, in the worst case. This means that there exists at least one source $\underline{X}$ with a joint probability distribution which is impossible to encode as if its components are independent without losing at least $\Theta(\log(m))$ bits. However, we now show that such sources are very "rare".

E. Average-Case Independent Representation

In this section we show that the expected value of $C(\underline{p}, g_{opt})$ is bounded by a small constant, when averaging uniformly over all possible $\underline{p}$ over an alphabet size m.

To prove this, we recall that $C(\underline{p}, g_{opt}) \leq C(\underline{p}, g_{ord})$ for any given probability distribution $\underline{p}$. Therefore, we would like to find the expectation of $C(\underline{p}, g_{ord})$ where the random variables are $p_1, \dots, p_m$, taking values over a uniform simplex.

Proposition 1: Let $\underline{X} \sim \underline{p}$ be a random vector of an alphabet size m and a joint probability distribution $\underline{p}$. The expected joint entropy of $\underline{X}$, where the expectation is over a uniform simplex of joint probability distributions $\underline{p}$ is

$$\mathbb{E}_{\underline{p}}\left\{H(\underline{X})\right\} = \frac{1}{\log_e 2}\left(\psi(m+1) - \psi(2)\right)$$

where ψ is the *digamma function*.

The proof of this proposition is left for the Appendix.

We now turn to examine the expected sum of the marginal entropies, $\sum_{j=1}^{d} H_b(Y_j)$ under the order permutation. As described above, the order permutation suggests sorting the probability distribution $p_1, \dots, p_m$ in an ascending order, followed by mapping of the i^{th} symbol (in a binary representation) the i^{th} smallest probability. Let us denote $p_{(1)} \leq \cdots \leq p_{(m)}$ the ascending ordered probabilities $p_1, \dots, p_m$. Bairamov *et al.* [23] show that the expected value of $p_{(i)}$ is

$$\mathbb{E}\left\{p_{(i)}\right\} = \frac{1}{m}\sum_{k=m+1-i}^{m}\frac{1}{k} = \frac{1}{m}\left(K_m - K_{m-i}\right) \qquad (13)$$

where $K_m = \sum_{k=1}^{m}\frac{1}{k}$ is the Harmonic number. Denote the ascending ordered binary representation of all possible symbols in a matrix form $A \in \{0,1\}^{(m \times d)}$. This means that entry A_{ij} corresponds to the j^{th} bit in the i^{th} symbol, when the symbols are given in an ascending order. Therefore, the expected sum of the marginal entropies of $\underline{Y}$, when the expectation is over a uniform simplex of joint probability distributions p, follows

$$\begin{aligned}\mathbb{E}_{\underline{p}}\left\{\sum_{j=1}^{d} H_b(Y_j)\right\} &\underset{(a)}{\leq} \sum_{j=1}^{d} h_b(\mathbb{E}_{\underline{p}}\{Y_j\}) \\ &\underset{(b)}{=} \sum_{j=1}^{d} h_b\left(\frac{1}{m}\sum_{i=1}^{m} A_{ij}\left(K_m - K_{m-i}\right)\right) \\ &\underset{(c)}{=} \sum_{j=1}^{d} h_b\left(\frac{1}{2}K_m - \frac{1}{m}\sum_{i=1}^{m} A_{ij}K_{m-i}\right) \qquad (14)\end{aligned}$$

where (a) follows from Jensen's inequality, (b) follows from (13) and (c) follows $\sum_{i=1}^{m} A_{ij} = \frac{1}{2}$ for all $j = 1, \dots, d$.

We now turn to derive asymptotic bounds of the expected difference between the sum of $\underline{Y}$'s marginal entropies and the joint entropy of $\underline{X}$, as appears in (8).

Theorem 2: Let $\underline{X} \sim \underline{p}$ be a random vector of an alphabet size m and joint probability distribution $\underline{p}$. Let $\underline{Y} = g_{ord}(\underline{X})$ be the order permutation. For $d \geq 10$, the expected value of $C(\underline{p}, g_{ord})$, over a uniform simplex of joint probability distributions $\underline{p}$, satisfies

$$\begin{aligned}&\mathbb{E}_{\underline{p}} C(\underline{p}, g_{ord}) \\ &\quad = \mathbb{E}_{\underline{p}}\left\{\sum_{j=1}^{d} H_b(Y_j) - H(\underline{X})\right\} < 0.0162 + O\left(\frac{1}{m}\right)\end{aligned}$$

Proof: Let us first derive the expected marginal entropy of the least significant bit, $j = 1$, according to (14).

$$\begin{aligned}\mathbb{E}_{\underline{p}}\{H_b(Y_1)\} &\leq h_b\left(\frac{1}{2}K_m - \frac{1}{m}\sum_{i=1}^{m/2} K_{m-i}\right) \\ &= h_b\left(\frac{1}{2}K_m - \frac{1}{m}\left(\sum_{i=1}^{m-1} K_i - \sum_{i=1}^{\frac{m}{2}-1} K_i\right)\right) \\ &\underset{(a)}{=} h_b\left(\frac{1}{2}K_m - \frac{1}{m}\left(mK_m - m - \frac{m}{2}K_{\frac{m}{2}} + \frac{m}{2}\right)\right) \\ &= h_b\left(\frac{1}{2}\left(K_{\frac{m}{2}} - K_m + 1\right)\right) \\ &\underset{(b)}{<} h_b\left(\frac{1}{2}\log_e\left(\frac{1}{2}\right) + \frac{1}{2} + O\left(\frac{1}{m}\right)\right) \\ &\underset{(c)}{\leq} h_b\left(\frac{1}{2}\log_e\left(\frac{1}{2}\right) + \frac{1}{2}\right) \\ &\qquad + O\left(\frac{1}{m}\right) h_b'\left(\frac{1}{2}\log_e\left(\frac{1}{2}\right) + \frac{1}{2}\right) \\ &= h_b\left(\frac{1}{2}\log_e\left(\frac{1}{2}\right) + \frac{1}{2}\right) + O\left(\frac{1}{m}\right) \qquad (15)\end{aligned}$$

where (a) and (b) follow the harmonic number properties:

(a) $\sum_{i=1}^{m} K_i = (m+1)K_{m+1} - (m+1)$

(b) $\frac{1}{2(m+1)} < K_m - \log_e(m) - \gamma < \frac{1}{2m}$, where γ is the Euler-Mascheroni constant [24]

and (c) results from the concavity of the binary entropy.

Repeating the same derivation for different values of j, we attain

$$\begin{aligned}&\mathbb{E}_{\underline{p}}\left\{H_b(Y_j)\right\} \\ &\quad \leq h_b\left(\frac{1}{2}K_m - \frac{1}{m}\sum_{l=1}^{2^j-1}(-1)^{l+1}\sum_{i=1}^{l\frac{m}{2^j}} K_{m-i}\right) \\ &\quad = h_b\left(\frac{1}{2}K_m - \frac{1}{m}\sum_{l=1}^{2^j}(-1)^{l}\sum_{i=1}^{l\frac{m}{2^j}-1} K_i\right) \\ &\quad = h_b\left(\frac{1}{2}K_m - \frac{1}{m}\sum_{l=1}^{2^j}(-1)^{l}\left(l\frac{m}{2^j}K_{l\frac{m}{2^j}} - l\frac{m}{2^j}\right)\right) \\ &\quad < h_b\left(\sum_{i=1}^{2^j-1}(-1)^{i+1}\frac{i}{2^j}\log_e\left(\frac{i}{2^j}\right) + \frac{1}{2}\right) + O\left(\frac{1}{m}\right) \qquad (16)\end{aligned}$$

for all $j = 1, \dots, d$. We may now evaluate the sum of expected marginal entropies of $\underline{Y}$. For simplicity of derivation let us obtain $\mathbb{E}_{\underline{p}}\left\{H_b(Y_j)\right\}$ for $j = 1, \dots, 10$ according to (16) and upper bound $\mathbb{E}_{\underline{p}}\left\{H_b(Y_j)\right\}$ for $j > 10$ with $h_b\left(\frac{1}{2}\right) = 1$. This means that for $d \geq 10$ we have

$$\mathbb{E}_{\underline{p}}\left\{\sum_{j=1}^{d} H_b(Y_j)\right\} < \sum_{j=1}^{10} \mathbb{E}_{\underline{p}}\left(H_b\left\{Y_j\right\}\right) + \sum_{j=11}^{d} h_b\left(\frac{1}{2}\right) < 9.4063 + (d-10) + O\left(\frac{1}{m}\right). \quad (17)$$

The expected joint entropy may also be expressed in a more compact manner. In Proposition 1 it is shown than $\mathbb{E}_{\underline{p}}\left\{H(\underline{X})\right\} = \frac{1}{\log_e 2}\left(\psi(m+1) - \psi(2)\right)$. Following the inequality in [24], the Digamma function, $\psi(m+1)$, is bounded from below by $\psi(m+1) = H_m - \gamma > \log_e(m) + \frac{1}{2(m+1)}$. Therefore, we conclude that for $d \geq 10$ we have that

$$\mathbb{E}_{\underline{p}}\left\{\sum_{j=1}^{d} H_b(Y_j) - H(\underline{X})\right\} < 9.4063 + (d-10) - \log(m) + \frac{\psi(2)}{\log_e 2} + O\left(\frac{1}{m}\right) = 0.0162 + O\left(\frac{1}{m}\right) \quad (18)$$

■

In addition, we would like to evaluate the expected difference between the sum of marginal entropies and the joint entropy of $\underline{X}$, that is, without applying any permutation. This serves us as a reference to the upper bound we achieve in Theorem 2.

Theorem 3: Let $\underline{X} \sim \underline{p}$ be a random vector of an alphabet size m and joint probability distribution $\underline{p}$. The expected difference between the sum of marginal entropies and the joint entropy of $\underline{X}$, when the expectation is taken over a uniform simplex of joint probability distributions $\underline{p}$, satisfies

$$\mathbb{E}_{\underline{p}}\left\{\sum_{j=1}^{d} H_b(X_j) - H(\underline{X})\right\} < \frac{\psi(2)}{\log_e 2} = 0.6099$$

Proof: We first notice that $P\left(X_j = 1\right)$ equals the sum of one half of the probabilities $p_i, i = 1, \ldots, m$ for every $j = 1 \ldots d$. Assume p_i's are randomly (and uniformly) assigned to each of the m symbols. Then, $\mathbb{E}\{P\left(X_j = 1\right)\} = \frac{1}{2}$ for every $j = 1 \ldots d$. Hence,

$$\mathbb{E}_{\underline{p}}\left\{\sum_{j=1}^{d} H_b(X_j) - H(\underline{X})\right\} = \sum_{j=1}^{d} \mathbb{E}_{\underline{p}}\left\{H_b(X_j)\right\} - \mathbb{E}_{\underline{p}}\{H(\underline{X})\} < d - \log(m) + \frac{1}{\log_e 2}\left(\psi(2) - \frac{1}{2(m+1)}\right) < \frac{\psi(2)}{\log_e 2}$$

■

To conclude, we show that for a random vector $\underline{X}$ over an alphabet size m, we have

$$\mathbb{E}_{\underline{p}} C(\underline{p}, g_{opt}) \leq \mathbb{E}_{\underline{p}} C(\underline{p}, g_{bst}) \leq \mathbb{E}_{\underline{p}} C(\underline{p}, g_{ord}) < 0.0162 + O\left(\frac{1}{m}\right) \quad (19)$$

for $d \geq 10$, where the expectation is over a uniform simplex of joint probability distributions $\underline{p}$. This means that when the alphabet size is large enough, even the simple order permutation achieves, on the average, a sum of marginal entropies which is only 0.0162 bits greater than the joint entropy, when all possible probability distributions $\underline{p}$ are equally likely to appear. Moreover, we show that the simple order permutation reduced the expected difference between the sum of the marginal entropies and the joint entropy of $\underline{X}$ by more than half a bit, for sufficiently large m.

Notice that the uniform prior assumption may not be adequate for every setup. In fact, this assumption is justified when we "do not know anything" about the distribution we are to encounter, so all distributions are considered equally likely to appear. Obviously, different families of distributions may be considered. For example, if we assume that $\underline{p}$ is the outcome of a Markov process, then it is easy to show that many of its values p_i are equal. This leads to a reduced complexity when applying our suggested algorithms, as less values need to be considered. Further discussion regarding parametric families of $\underline{p}$ and their theoretical and computational implications are provided in [18].

V. Large Alphabet Source Coding

Assume a classic compression setup in which both the encoder and the decoder are familiar with the joint probability distribution of the source $\underline{X} \sim \underline{p}$, and the number of observations n is sufficiently large in the sense that $\hat{H}(\underline{X}) \approx H(\underline{X})$. As discussed above, both Huffman and arithmetic coding entail a growing redundancy and a quite involved implementation as the alphabet size increases. The Huffman code guarantees a redundancy of at most a single bit for every alphabet size, depending on the dyadic structure of p. On the other hand, arithmetic coding does not require a dyadic p, but only guarantees a redundancy of up to two bits, and is practically limited for a smaller alphabet size [3], [8]. In other words, both Huffman and arithmetic coding are quite likely to have an average codeword length which is greater than $H(\underline{X})$, and are complicated (or sometimes even impossible) to implement, as the alphabet size increases.

To overcome these drawbacks, we suggest a simple solution in which we first apply an invertible transformation to make the components of $\underline{X}$ "as statistically independent as possible", followed by entropy coding to each of its components separately. This scheme results in an overhead cost of $C(\underline{p}, g) = \sum_{j=1}^{d} H(Y_j) - H(\underline{X})$. However, it allows us to apply a Huffman or arithmetic encoding on each of the components separately; hence, over a binary alphabet.

Moreover, notice we can group several components, Y_j, into blocks so that the joint entropy of the block is necessarily lower than the sum of marginal entropies of Y_j. Specifically, denote b as the number of components in each block and B as the number of blocks. Then, $b \times B = d$ and for each block $\upsilon = 1, \ldots, B$ we have that

$$H(\underline{Y}^{(\upsilon)}) \leq \sum_{u=1}^{b} H_b(Y_u^{(\upsilon)}) \quad (20)$$

where $H(\underline{Y}^{(\upsilon)})$ is the entropy of the block υ and $H_b(Y_u^{(\upsilon)})$ is the marginal entropy of the u^{th} component of the block υ. Summing over all B blocks we have

$$\sum_{\upsilon=1}^{B} H(\underline{Y}^{(\upsilon)}) \leq \sum_{\upsilon=1}^{B}\sum_{u=1}^{b} H_b(Y_u^{(\upsilon)}) = \sum_{j=1}^{d} H_b(Y_j). \qquad (21)$$

This means we can always apply our suggested invertible transformation which minimizes $\sum_{j=1}^{d} H_b(Y_j)$, and then the group components into B blocks and encode each block separately. This results in $\sum_{\upsilon=1}^{B} H(\underline{Y}^{(\upsilon)}) \leq \sum_{j=1}^{d} H_b(Y_j)$. By doing so, we increase the alphabet size of each block (to a point which is still not problematic to implement with Huffman or arithmetic coding) while at the same time we decrease the redundancy results from $\sum_{j=1}^{d} H_b(Y_j) - H(\underline{X})$. We discuss different considerations in choosing the number of blocks B in the following sections.

A more direct approach of minimizing the sum of block entropies $\sum_{\upsilon=1}^{B} H(\underline{Y}^{(\upsilon)})$ is to refer to each block as a symbol over a greater alphabet size, 2^b. This allows us to seek an invertible transformation which minimizes the sum of marginal entropies, where each marginal entropy corresponds to a marginal probability distribution over an alphabet size 2^b. This minimization problem is referred to as a generalized ICA over finite alphabets and is discussed in detail in [18].

However, notice that both the Piece-wise Linear Relaxation algorithm (Section IV-B), and the solutions discussed in [18], require an extensive computational effort in finding a minimizer for (8) as the alphabet size increases. Therefore, we suggest applying the greedy order permutation as d grows. This solution may result in quite a large value of $C(\underline{p}, g_{ord})$ for several joint probability distributions $\underline{p}$ (as shown in Section IV-D). However, as we uniformly average over all possible p's, the redundancy is bounded by a small constant as the alphabet size increases (Section IV-E). Moreover, the ordering approach simply requires ordering the values of $\underline{p}$, which is significantly faster than constructing a Huffman dictionary or arithmetic encoder.

To illustrate our suggested scheme, consider a source $\underline{X} \sim \underline{p}$ over an alphabet size 2^d, which follows the Zipf's law distribution,

$$P(k; s, 2^d) = \frac{k^{-s}}{\sum_{l=1}^{2^d} l^{-s}}$$

where s is the "skewness" parameter. The Zipf's law distribution is a commonly used heavy-tailed distribution, mostly in modeling of natural (real-world) quantities. It is widely used in physical and social sciences, linguistics, economics and many other fields. We would like to design an entropy code for $\underline{X}$, for $d = 16$, and over different values of s. We apply a standard Huffman code as an example of a common entropy coding scheme. We further apply our suggested order permutation scheme (Section IV-C), in which we sort $\underline{p}$ in a descending order, followed by arithmetic encoding (as the alphabet is now smaller) to each of the components separately. Then, we group these components into two separate blocks (as discussed above) and apply an arithmetic encoder on each of the blocks. We repeat this experiment for a range of parameter values s. Figure 2 demonstrates the results we achieve. Our results show that the Huffman code attains an average codeword length which is very close to the entropy of the source for lower values of s. However, as s increases and the distribution of the source becomes more skewed, the Huffman code diverges from the entropy. On the other hand, our suggested method succeeds in attaining an average codeword length which is very close to the entropy of the source for every s, especially as s increases and when independently encoding each of the blocks.

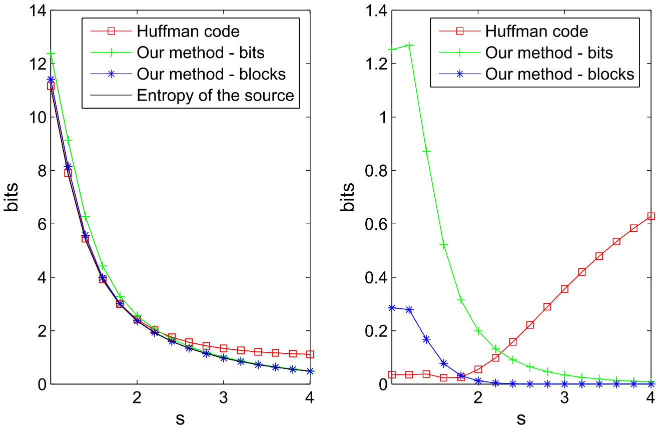

Fig. 2. Zipf's law simulation results. Left: the curve with the squares is the average codeword length using a Huffman code, the curve with the crosses corresponds to the average codeword length using our suggested methods when encoding each component separately, and the curve with the asterisks is our suggested method when encoding each of the two blocks separately. The black curve (which tightly lower-bounds all the curves) is the entropy of the source. Right: The difference between each encoding method and the entropy of the source.

VI. Universal Source Coding

The classical source coding problem is typically concerned with a source whose alphabet size is much smaller than the length of the sequence. In this case one usually assumes that $\hat{H}(\underline{X}) \approx H(\underline{X})$. However, in many real world applications such an assumption is invalid. A paradigmatic example is the word-wise compression of natural language texts. In this setup we draw a memoryless sequence of words, so that the alphabet size is often comparable to or even larger than the length of the source sequence.

As discussed above, the main challenge in large alphabet source coding is the redundancy of the code, which is formally defined as the excess number of bits used over the source's entropy. In section III we describe a variety of worst-case redundancy results, such as [10]–[12] and others.

Here, we claim again that in some cases (not necessarily the worst) applying a transformation which decomposes the observed sequence into multiple "as independent as possible" components results in a lower redundancy than currently known practical methods. Notice that encoding each component separately brings us back to the classical "small alphabet" regime, as the components are binary while the sequence length remains the same. This means that as in the previous section, applying BICA may overcome the large alphabet problem, in the cost of the remaining dependency between the components (7). However we realize that now the encoder needs to transmit not only the encoded sequence and the code, but also the transformation that we apply.

Assume that we apply the order permutation to encode a sequence of length n over an alphabet size 2^d, where $2^d > n$. Then, describing the transformation requires transmitting at most n unique order values, each of length d bits. This means that the order permutation redundancy is alone $n \cdot d$ bits, which is very costly and technically impractical.

Therefore, we require a different approach which, as before, minimizes the sum of marginal entropies (8), but at the same time is less costly to describe. Since any general transformation that is a one-to-one mapping of 2^d words would require at least $n \cdot d$ bits to describe, we need to consider more "compact" transformations, that would unfortunately be less effective in minimizing (8).

One possible solution is to seek for linear (and invertible) transformations. Since every word in the alphabet is described by d bits, a linear transformation is simply a squared matrix of d^2 digits. This means that describing a linear transformation would only require d^2 bits. Unfortunately, the linear BICA problem is quite involved and not robust to a growing alphabet size. In their work, Attux *et al.* [25] describe the difficulties in minimizing (8) over the XOR field (linear transformations) and suggest an immune-inspired algorithm for it. Their algorithm, which is a heuristic in its essence, demonstrates some promising results. However, it is not scalable and preforms quite poorly as a minimizer to (8) when d, the dimension of the problem, increases. Moreover, it does not guarantee to converge to the global optimal solution.

Therefore, we would like to to take a different path and modify our suggested piece-wise linear approach (Section IV-B) so that the transformation we achieve requires fewer bits to describe.

The core difficulty in transmitting our previously described transformations is that we allow a one-to-one mapping of all 2^d words. Therefore, we propose a "block" oriented method, in which we split the original d components into B separate blocks (with b components in every block) and apply a transformation on each of these blocks independently. For simplicity assume that $B = \frac{d}{b}$ is a natural number. In other words, instead of encoding a d-bits vector (over an alphabet size of 2^d) , we shall encode B vectors of b bits separately (where each vector is over an alphabet size of only 2^b). Assuming that $n \gg 2^b$, communicating B transformations to the receiver takes only $B \cdot b \cdot 2^b$. In addition, since $n \gg 2^b$, we can regard the alphabet of each block as "small" compared with the sequence length, so that the worst-case redundancy of communicating the code of each of these blocks follows the case of $n = o(2^b)$, as appears in (2). Therefore, a block-wise compression would take about

$$n \cdot \sum_{v=1}^{B} \hat{H}(\underline{X}^{(v)}) + B\frac{2^b - 1}{2} \log \frac{n}{2^b} + B \cdot b \cdot 2^b \tag{22}$$

bits, where the first term is the cost of transmitting the B encoded sequences, the second term is B times the worst-case redundancy (for each of the blocks), and the third term is the cost of transmitting B transformations (as previously described). Notice that this block-wise encoding scheme implies additional degrees of freedom in its design. First (and obvious) is the value of b. Second, is the grouping of d components into B disjoint blocks.

Let us start by fixing b and focus on the grouping problem. A naive grouping approach is to exhaustively or randomly search for all possible combinations of grouping d components into B blocks. However, assuming that d is quite large, an exhaustive search is practically infeasible, while a random search is simply not good enough (as demonstrated below). Therefore, a different method is required.

As before, let $\underline{X}$ be the source vector while $\underline{Y}$ is the resulting vector, after we apply our suggested transformation. Every block of the vector $\underline{Y}$ satisfies (20), where the entropy terms are now replaced with empirical entropies. In the same manner as in Section V, summing over all B blocks results in (21) where again, the entropy terms are replaced with empirical entropies. This means that the sum of the empirical block entropies is bounded from above by the sum of empirical marginal entropies of the components of $\underline{Y}$ (with equality iff the components are independently distributed).

$$\sum_{v=1}^{B} \hat{H}(\underline{Y}^{(v)}) \leq \sum_{j=1}^{d} \hat{H}_b(Y_j). \tag{23}$$

Our suggested scheme works as follows: We first randomly partition the d components into B blocks. Then, we apply the generalized BICA on each block. The sum of empirical marginal entropies (of each block) is an upper bound on the empirical entropy of each block, as described in the previous paragraph. Now, let us randomly shuffle the d components of the vector $\underline{Y}$. By "shuffle" we refer to an exchange of positions of $\underline{Y}$'s components. Notice that by doing so, the sum of empirical marginal entropies of the entire vector $\sum_{i=1}^{d} \hat{H}_b(Y_i)$ is maintained. We now apply the generalized BICA on each of the (new) blocks. This way we minimize (or at least do not increase) the sum of empirical marginal entropies of each of the (new) blocks. This obviously results in a lower sum of empirical marginal entropies for the entire vector $\underline{Y}$. It also means that we minimize the right hand side of (23), which bounds from above the sum of empirical block entropies, as this inequality suggests. In other words, in each iteration we decrease (at least do not increase) an upper bound on our objective. We terminate once a maximal number of iterations is reached or when we can no longer decrease the sum of empirical marginal entropies. Assuming we terminate at iteration I_0, encoding the data takes about

$$n \cdot \sum_{v=1}^{B} \hat{H}^{[I_0]}(\underline{Y}^{(v)}) + B\frac{2^b - 1}{2} \log \frac{n}{2^b} + I_0 B \cdot b \cdot 2^b + I_0 d \log d \tag{24}$$

bits, where the first term refers to the sum of empirical block entropies at the I_0 iteration, the second term is the redundancy of the B codes (for each of the blocks), the third term refers to the representation of $I_0 \cdot B$ invertible transformation of each block during the process until I_0, and the fourth term refers to the bit shuffling at the beginning of each iteration. Hence,

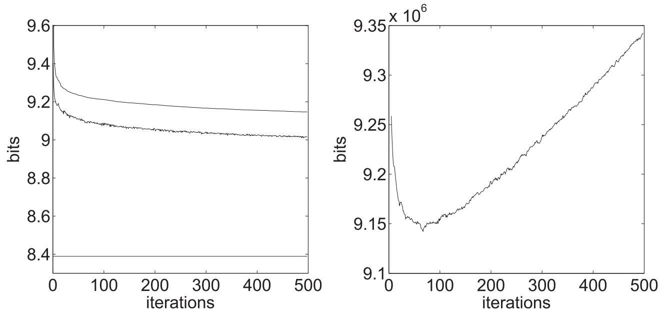

Fig. 3. Large Alphabet Source Coding via Generalized BICA with $B = 4$ blocks. Left side: the horizontal line indicated the empirical entropy of $\underline{X}$. The upper curve is the sum of marginal empirical entropies and the lower curve is the sum of empirical block entropies (the outcome of our suggested framework). Right side: total compression size of our suggested method at each iteration.

in order minimize (24) we need to find the optimal trade-off between a low value of $\sum_{v=1}^{B} \hat{H}^{[I_0]}(\underline{Y}^{(v)})$ and a low iteration number I_0. Finally, we may apply this technique with different values of b to find the best compression scheme over all block sizes.

A. Synthetic Experiments

We begin the demonstration of our suggested method by generating a data-set according to the Zipf law distribution that was previously described. In this experiment we discuss the best way to represent this data-set over a binary alphabet, in the sense of a minimal compression rate. We draw $n = 10^6$ independent realizations from this distribution with an alphabet size $m = 2^{20}$ and a parameter value $s = 1.2$. We encounter $n_0 = 80\,071$ unique words and attain an empirical entropy of 8.38 bits (while the true entropy is 8.65 bits). Therefore, compressing the drawn realizations in its given 2^{20} alphabet size takes a total of about $10^6 \times 8.38 + 1.22 \times 10^6 = 9.6 \cdot 10^6$ bits, according to (4). Using the patterns method [13], the redundancy we achieve is the redundancy of the pattern plus the size of the dictionary. Hence, the compressed size of the data set according to this method is bounded from below by $10^6 \times 8.38 + 80\,071 \times 20 + 100 = 9.982 \cdot 10^6$ bits. In addition to these asymptotic schemes we would also like to compare our method with a common practical approach. For this purpose we apply the canonical version of the Huffman code. Through the canonical Huffman code we are able to achieve a compression rate of 9.17 bits per symbol, leading to a total compression size of about $1.21 \cdot 10^7$ bits.

Let us now apply a block-wise compression. We first demonstrate the behavior of our suggest approach with four blocks ($B = 4$) as appears in Figure 3. To have a good starting point, we initiate our algorithm with a naive random shuffle search over the components (described above). The plot on the left demonstrates the sum of marginal empirical entropies and the sum of empirical block entropies as the number of iterations increases. As we can see in the plot on the right, our suggested approach achieves a minimal compression size of $9.144 \cdot 10^6$ bits, when $I_0 = 64$.

TABLE I

Block-Wise Compression Via Generalized BICA Method for Different Block Sizes

B	Min. of $\sum \hat{H}(\underline{Y}^{(v)})$	Opt. I_0	Compressed Data Size	Redund.	Total Size
2	8.69	5	$8.69 \cdot 10^6$	$1.15 \cdot 10^5$	$\mathbf{8.805 \cdot 10^6}$
3	8.93	19	$8.93 \cdot 10^6$	$5.55 \cdot 10^4$	$8.985 \cdot 10^6$
4	9.09	64	$9.09 \cdot 10^6$	$5.41 \cdot 10^4$	$9.144 \cdot 10^6$

Table I summarizes the results we achieve for different block sizes B. We see that the lowest compression size is when $B = 2$, i.e. two blocks. The reason is that we earn a great coding redundancy reduction (from large to "not large" alphabet) already when turning to a two-block representation, while not losing too much in terms of minimizing (8) and the cost of transmitting the two block transformations. We further notice that the optimal iterations number grows with the number of blocks. This results from the cost of describing the optimal transformation for each block, at each iteration, $I_0 B \cdot b \cdot 2^b$, which exponentially increases with the block size b. Comparing our results with the three methods described above we are able to reduce the total compression size in $8 \cdot 10^5$ bits, compared to the minimum among all of our competitors.

B. Real-World Experiments

We now demonstrate our suggested compression framework on real world data-sets. For this purpose we use collections of word frequencies of different natural languages. These word frequency lists are publicly available[1] and describe the frequency each word appears in a language, based on hundreds of millions of words, collected from open source subtitles[2] or based on different dictionaries and glossaries [26]. Since each word frequency list holds several hundreds of thousands of different words, we choose a binary $d = 20$ bit representation. We sample $n = 10^7$ words from each language and examine our suggested framework, compared with the compression schemes mentioned above. The results we achieve are summarized in Table II. Notice the last column provides the percentage redundancy reduction, which is essentially the most we can hope for (as we cannot go under $n \cdot \hat{H}(\underline{X})$ bits). As in the previous experiment, our suggested algorithm achieves the lowest compression size applied with two blocks after approximately $I_0 = 10$ iterations, from the same reasons mentioned above. Compared with the other methods, our suggested framework shows to achieve significantly lower compression sizes for all languages, saving an average of over one million bits per language.

VII. Vector Quantization

Vector quantization refers to a lossy compression setup, in which a high-dimensional vector $\underline{X} \in \mathbb{R}^d$ is to be represented by a finite number of points. This means that the high dimensional observed samples are clustered into groups, where

[1] http://en.wiktionary.org/wiki/Wiktionary:Frequency_lists

[2] www.opensubtitles.org

TABLE II

NATURAL LANGUAGES EXPERIMENT. FOR EACH COMPRESSION METHOD (D), (O) AND (T) STAND FOR THE COMPRESSED DATA, THE OVERHEAD AND THE TOTAL COMPRESSION SIZE (IN BITS) RESPECTIVELY. THE WE SAVE COLUMN IS THE AMOUNT OF BITS SAVED BY OUR METHOD, AND ITS CORRESPONDING PERCENTAGE OF (O) AND (T). n_0 IS THE NUMBER OF UNIQUE WORDS OBSERVED IN EACH LANGUAGE, OF THE 10^7 SAMPLED WORDS. NOTICE THE CHINESE CORPUS REFERS TO CHARACTERS

Language (n_0)	Standard Compression	Patterns Compression	Canonical Huffman	Our Suggested Method	We Save
English $(129,834)$	(D) $9.709 \cdot 10^7$ (O) $2.624 \cdot 10^6$ (T) $9.971 \cdot 10^7$	(D) $9.709 \cdot 10^7$ (O) $2.597 \cdot 10^6$ (T) $9.968 \cdot 10^7$	(D) $9.737 \cdot 10^7$ (O) $5.294 \cdot 10^6$ (T) $1.027 \cdot 10^8$	(D) $9.820 \cdot 10^7$ (O) $2.207 \cdot 10^5$ (T) $\mathbf{9.842 \cdot 10^7}$	$1.262 \cdot 10^6$ (O) 48.6% (T) 1.27%
Chinese $(87,777)$	(D) $1.020 \cdot 10^8$ (O) $2.624 \cdot 10^6$ (T) $1.046 \cdot 10^8$	(D) $1.020 \cdot 10^8$ (O) $1.696 \cdot 10^6$ (T) $1.037 \cdot 10^8$	(D) $1.023 \cdot 10^8$ (O) $3.428 \cdot 10^6$ (T) $1.057 \cdot 10^8$	(D) $1.028 \cdot 10^8$ (O) $2.001 \cdot 10^5$ (T) $\mathbf{1.030 \cdot 10^8}$	$6.566 \cdot 10^5$ (O) 38.7% (T) 0.63%
Spanish $(185,866)$	(D) $1.053 \cdot 10^8$ (O) $2.624 \cdot 10^6$ (T) $1.079 \cdot 10^8$	(D) $1.053 \cdot 10^8$ (O) $3.718 \cdot 10^6$ (T) $1.090 \cdot 10^8$	(D) $1.055 \cdot 10^8$ (O) $7.700 \cdot 10^6$ (T) $1.132 \cdot 10^8$	(D) $1.067 \cdot 10^8$ (O) $2.207 \cdot 10^5$ (T) $\mathbf{1.069 \cdot 10^8}$	$9.631 \cdot 10^5$ (O) 36.7% (T) 0.89%
French $(139,674)$	(D) $1.009 \cdot 10^8$ (O) $2.624 \cdot 10^6$ (T) $1.035 \cdot 10^8$	(D) $1.009 \cdot 10^8$ (O) $2.794 \cdot 10^6$ (T) $1.036 \cdot 10^8$	(D) $1.011 \cdot 10^8$ (O) $5.745 \cdot 10^6$ (T) $1.069 \cdot 10^8$	(D) $1.017 \cdot 10^8$ (O) $2.207 \cdot 10^5$ (T) $\mathbf{1.019 \cdot 10^8}$	$1.557 \cdot 10^6$ (O) 59.3% (T) 1.50%
Hebrew $(250,917)$	(D) $1.173 \cdot 10^8$ (O) $2.624 \cdot 10^6$ (T) $1.200 \cdot 10^8$	(D) $1.173 \cdot 10^8$ (O) $5.019 \cdot 10^6$ (T) $1.224 \cdot 10^8$	(D) $1.176 \cdot 10^8$ (O) $1.054 \cdot 10^7$ (T) $1.281 \cdot 10^8$	(D) $1.190 \cdot 10^8$ (O) $1.796 \cdot 10^5$ (T) $\mathbf{1.192 \cdot 10^8}$	$7.837 \cdot 10^5$ (O) 29.9% (T) 0.65%

each group is represented by a representative point. For example, the famous k-means algorithm [27] provides a method to determine the clusters and the representative points (centroids) for an Euclidean loss function. Then, these centroid points that represent the observed samples are compressed in a lossless manner.

As described above, in the lossy encoding setup one is usually interested in minimizing the amount of bits which represent the data for a given a distortion (or equivalently, minimizing the distortion for a given compressed data size). The rate-distortion function defines the lower bound on this objective. In vector quantization, the representation is a deterministic mapping (defined as $P(\underline{Y}|\underline{X})$) from a source $\underline{X}$ to its quantized version $\underline{Y}$. Therefore we have that $H(\underline{Y}|\underline{X}) = 0$ and the rate distortion is simply

$$R(D) = \min_{P(\underline{Y}|\underline{X})} H(\underline{Y}) \ s.t. \ \mathbb{E}\left\{D(\underline{X}, \underline{Y})\right\} \leq D \tag{25}$$

where $D(\underline{X}, \underline{Y})$ is some distortion measure between $\underline{X}$ and $\underline{Y}$.

A. Entropy Constrained Vector Quantization

The Entropy Constrained Vector Quantization (ECVQ) is an iterative method for clustering the observed samples into centroid points which are later represented by a minimal average codeword length. The ECVQ algorithm aims to find the minimizer of

$$J(D) = \min \mathbb{E}\left\{l(\underline{X})\right\} \ s.t. \ \mathbb{E}\left\{D(\underline{X}, \underline{Y})\right\} \leq D \tag{26}$$

where the minimization is over three terms: the vector quantizer (of $\underline{X}$), the entropy encoder (of the quantized version of $\underline{X}$) and the reconstruction module of $\underline{X}$ from its quantized version.

Let us use a similar notation to [28]. Denote the vector quantizer $\alpha : \underline{x} \rightarrow \mathcal{C}$ as a mapping from an observed sample to a cluster in $\mathcal{C}$, where $\mathcal{C}$ is a set of m clusters. Further, let $\gamma : \mathcal{C} \rightarrow c$ be a mapping from a cluster to a codeword. Therefore, the composition $\alpha \circ \gamma$ is the encoder. In the same manner, the decoder is a composition $\gamma^{-1} \circ \beta$, where γ^{-1} is the inverse mapping from a codeword to a cluster and $\beta : \mathcal{C} \rightarrow \underline{y}$ is the reconstruction of $\underline{x}$ from its quantized version. Therefore, the Lagrangian of the optimization problem (26) is

$$L_\lambda(\alpha, \beta, \gamma) = \mathbb{E}\left\{D(\underline{X}, \beta\left(\alpha\left(\underline{X}\right)\right) + \lambda\left|\gamma\left(\alpha\left(\underline{X}\right)\right)\right|\right\} \tag{27}$$

The ECVQ objective is to find the coder (α, β, γ) which minimizes this functional. Chou et al. [28] suggest an iterative descent algorithm similar to the generalized Lloyd algorithm [15]. Their algorithm starts with an arbitrary initial coder. Then, for a fixed γ and β it finds a clustering $\alpha(\underline{X})$ as the minimizer of:

$$\alpha(\underline{X}) = \operatorname*{argmin}_{i \in \mathcal{C}} \left\{D(\underline{X}, \beta(i)) + \lambda\left|\gamma(i)\right|\right\}. \tag{28}$$

Notice that for an Euclidean distortion, this problem is simply k-means clustering, with a "bias" of $\lambda|\gamma(i)|$ on its objective function.

For a fixed α and β, we notice that each cluster $i \in \mathcal{C}$ has an induced probability of occurrence p_i. Therefore, the entropy encoder γ is designed accordingly, so that $|\gamma(i)|$ is minimized. The Huffman algorithm could be incorporated into the design algorithm at this stage. However, for simplicity, allow the fiction that codewords can have non-integer lengths, and assign

$$|\gamma(i)| = -\log(p_i). \tag{29}$$

Finally, for a fixed α and γ, the reconstruction module β is

$$\beta(i) = \underset{\underline{y}\in\underline{Y}}{\operatorname{argmin}}\ \mathbb{E}\left\{D\left(\underline{X},\underline{y}\right)|\alpha(\underline{X})=i\right\}. \qquad (30)$$

For example, for an euclidean distortion measure, $\beta(i)$'s are simply the centroids of the clusters $i \in \mathcal{C}$.

Notice that the value objective (27), when applying each of the three steps (28-30), is non-increasing. Therefore, as we apply these three steps repeatedly, the ECVQ algorithm is guarenteed to converge to a local minimum. Moreover, notice that for an Euclidean distortion measure, step (28) of the ECVQ algorithm is a variant of the k-means algorithm. However, the k-means algorithms is known to be computationally difficult to execute as the number of observed samples increases. Hence, the ECVQ algorithm is also practically limited to a relatively small number of samples.

As in previous sections, we argue that when the alphabet size is large (corresponds to low distortion), it may be better to encode the source component-wise. This means, we would like to construct a vector quantizer such that the sum marginal entropies of $\underline{Y}$ is minimal, subject to the same distortion constraint as in (25). Specifically,

$$\tilde{R}(D) = \min_{P(\underline{Y}|\underline{X})} \sum_{j=1}^{d} H_b(Y_j)\ s.t.\ \mathbb{E}\left\{D(\underline{X},\underline{Y})\right\} \le D \qquad (31)$$

Notice that for a fixed distortion value, $R(D) \le \tilde{R}(D)$ as sum of marginal entropies is bounded from below by the joint entropy. However, since encoding a source over a large alphabet may result in a large redundancy (as discussed in previous sections), the average codeword length of the ECVQ (26) is not necessarily lower than our suggested method (and usually even much larger).

Our suggested version of the ECVQ works as follows: we construct α and β in the same manner as ECVQ does, but replace the Huffman encoder (in γ) with our suggested relaxation to the BICA problem (Section IV-B). This means that for a fixed α, β, which induce a random vector over a finite alphabet size (with a finite probability distribution), we seek for a representation which makes its components "as statistically independent as possible". The average codeword lengths are then achieved by arithmetic encoding on each of these components.

This scheme results not only in a different codebook, but also with a different quantizer than the ECVQ. This means that a quantizer which strives to construct a random vector (over a finite alphabet) with the lowest possible average codeword length (subject to a distortion constraint) is different than our quantizer, which seeks for a random vector with a minimal sum of marginal average codeword lengths (subject to the same distortion).

Our suggested scheme proves to converge to a local minimum in the same manner that ECVQ does. That is, for a fixed α, β, our suggested relaxed BICA method finds a binary representation which minimizes the sum of marginal entropies. Therefore, we can always compare the representation it achieves in the current iteration with the representation it found in the previous iteration, and choose the one which minimizes the objective. This leads to a non-increasing objective each time it is applied. Moreover, notice that we do not have to use the complicated relaxed BICA scheme and apply the simpler order permutation (Section IV-C). This would only result in a possible worse encoder but local convergence is still guaranteed.

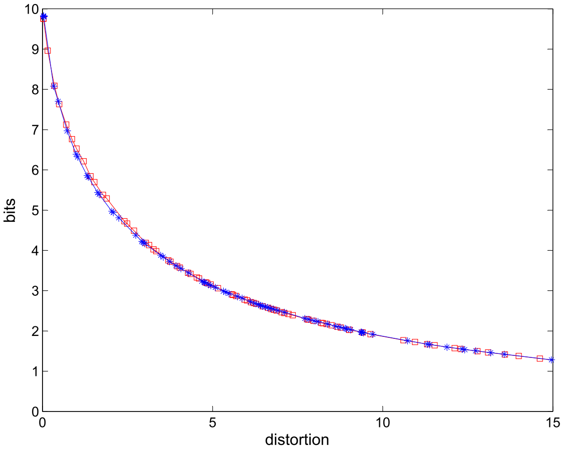

Fig. 4. ECVQ simulation. The curve with the squares corresponds to the average codeword length achieved by the classical ECVQ algorithm. The curve with the asterisks is the average codeword length achieved by our suggested BICA variant to the ECVQ algorithm.

To illustrate the performance of our suggested method we conduct the following experiment: We draw 1000 independent samples from a six dimensional bivariate Gaussian mixture. We apply both the ECVQ algorithm, and our suggest BICA variation of the ECVQ, on these samples. Figure 4 demonstrates the average codeword length we achieve for different Euclidean (mean square error) distortion levels.

We first notice that both methods performs almost equally well. The reason is that 1000 observations do not necessitate an alphabet size which is greater than $m = 1000$ to a attain a zero distortion. In this "small alphabet" regime, our suggested approach does not demonstrate its advantage over classical methods, as discussed in previous sections. However, we can still see it performs equally well.

As we try to increase the number of observations (and henceforth the alphabet size) we encounter computational difficulties, which result from repeatedly performing a variant of the k-means algorithm (28). This makes both ECVQ and our suggested method quite difficult to implement over a "large alphabet size" (many observations and low distortion). However, notice that if Gersho's conjecture [29] is true, and the best space-filling polytope is a lattice, then the optimal d-dimensional ECVQ at high resolution (low distortion) regime takes the form of a lattice [30]. This means that for this setup, γ is simply a lattice quantizer. This idea is discussed in greater detail in the next section.

B. Vector Quantization With Fixed Lattices

As demonstrated in the previous section, applying the ECVQ algorithm to a large number of observations with a low distortion constraint is impractical. To overcome this problem we suggest using a predefined quantizer in the form of a lattice. This means that instead of seeking for a quantizer γ that results in a random vector (over a finite alphabet) with a low average codeword length, we use a fixed quantizer, independent of the samples, and construct a codebook accordingly. Therefore, the performance of the codebook strongly depends on the empirical entropy of the quantized samples. Since we are dealing with fixed lattices (vector quantizers), it is very likely that the empirical entropy of the quantized samples would be significantly different (lower) than the true entropy in low distortion regimes (large alphabet size). Therefore, the compressed data would consist of both the compressed samples themselves and a redundancy term, as explained in detail in Section VI.

Here again, we suggest that instead of encoding the quantized samples over a large alphabet size, we should first represent them with components that are "as statistically independent as possible", and encode each component separately. To demonstrate this scheme we turn to a classic quantizing problem, of a standard d-dimensional normal distribution. Notice this quantizing problem is very well studied [3] and a lower bound for the average codeword length, for a given distortion value D, is given by

$$R(D) = \max\left\{\frac{d}{2}\log\left(\frac{d}{D}\right), 0\right\}. \tag{32}$$

In this experiment we draw n samples from a standard d-dimensional multivariate normal distribution. Since the span of the normal distribution is infinite, we use a lattice which is only defined in a finite sphere. This means that each sample which falls outside this sphere is quantized to its nearest quantization point on the surface of the sphere. We define the radius of the sphere to be 5 times the variance of the source (hence $r = 5$). We first draw $n = 10^5$ samples from $d = 3, 4$ and 8 dimensional normal distributions. For $d = 3$ we use a standard cubic lattice, while for $d = 4$ we use an hexagonal lattice [30]. For $d = 8$ we use an 8-dimensional integer lattice [30]. The upper row of Figure 5 demonstrates the results we achieve for the three cases respectively (left to right), where for each setup we compare the empirical joint entropy of the quantized samples (dashed line) with the sum of empirical marginal entropies, following our suggested approach (solid line). We further indicate the rate distortion lower bound (32) for each scenario, calculated according to the true distribution (line with x's). Notice the results are normalized according to the dimension d. As we can see, the sum of empirical marginal entropies is very close to the empirical joint entropy for $d = 3, 4$. The rate distortion indeed bounds from below both of these curves. For $d = 8$ the empirical joint entropy is significantly lower than the true entropy (especially in the low distortion regime). This is a result of an alphabet size which is larger than the number of samples n. However, in this case too, the sum of empirical marginal entropies is close to the joint empirical entropy. The behavior described above is maintained as we increase the number of samples to $n = 10^6$, as indicated in the lower row of Figure 5. Notice again that the sum of marginal empirical entropies is very close to the joint empirical entropy, especially on the bounds (very high and very low distortion). The reason is that in both of these cases, where the joint probability is either almost uniform (low distortion) or almost degenerate (high distortion), there exists a representation which makes the components statistically independent. In other words, both the uniform and degenerate distributions can be shown to satisfy $\sum_{j=1}^{d} H_b(Y_j) = H(\underline{Y})$ using the order permutation.

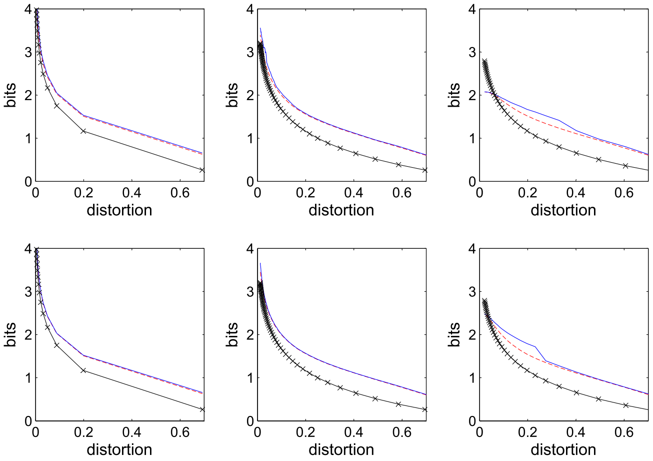

Fig. 5. Lattice quantization of d-dimensional standard normal distribution. The upper row corresponds to $n = 10^5$ drawn samples while the lower row is $n = 10^6$ samples. The columns correspond to the dimensions $d = 3, 4$ and 8 respectively. In each setup, the dashed line is the joint empirical entropy while the solid line is the sum of marginal empirical entropies, following our suggested method. The line with the x's is the rate distortion (32), calculated according to the true distribution.

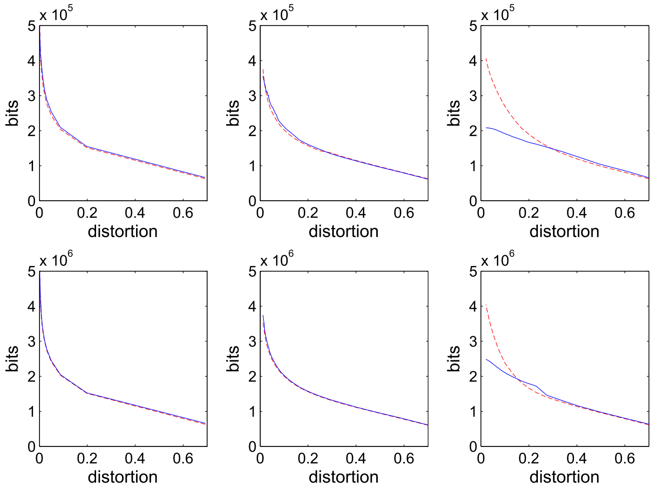

Fig. 6. Total compression size for lattice quantization of d-dimensional standard normal distribution. The upper row corresponds to $n = 10^5$ drawn samples while the lower row is $n = 10^6$ samples. The columns correspond to the dimensions $d = 3, 4$ and 8, from left to right. In each setup, the dashed line is the total compression size through classical universal compression while the solid line is the total compression size using our suggested relaxed generalized BICA approach.

We further present the total compression size of the quantized samples in this universal setting. Figure 6 shows the amount of bits required for the quantized samples, in addition to the overhead redundancy, for both Huffman coding and our suggested scheme. As before, the rows correspond to $n = 10^5$ and $n = 10^6$ respectively, while the columns are $d = 3, 4$ and 8, from left to right. We first notice that for $d = 3, 4$ both methods perform almost equally well. However, as d increases, there exists a significant different between the classical coding scheme and our suggested method, for low distortion rate. The reason is that for larger dimensions and low distortion rate, we need a very large number of quantization points, hence, a large alphabet size. This is exactly the regime where our suggested method demonstrates its enhanced capabilities, compared with standard methods.

VIII. Conclusions

In this work we introduce a conceptual framework for large alphabet source coding. We suggest to decompose a large alphabet source into components which are "as statistically independent as possible" and then encode each component separately. This way we overcome the well known difficulties of large alphabet source coding, at the cost of:

(i) The components not being perfectly independent
(ii) A computational difficulty in finding a transformation which decomposes the source.

We propose two methods which focus on minimizing these costs. The first method is a piece-wise linear relaxation to the BICA (Section IV-B). This method strives to decrease (i) as much as possible, but its computationally complexity is quite involved. Our second method is the order permutation (Section IV-C) which is very simple to implement (hence, focuses on (ii)) but results in a larger redundancy as it is a greedy solution to (7).

We show that while not every source can be efficiently decomposed into independent components, the vast majority of sources do decompose very well (that is, with only a small constant term) as the alphabet size increases. More specifically, we show that the average difference between the sum of marginal entropies (after the "simpler" order permutation is applied) and the joint entropy of the source is bounded by a small constant, as m increases. This means that even the order permutation, which is inferior to the relaxed BICA method, is capable of achieving a very low overhead cost for many large alphabet sources.

We demonstrate our suggested framework on three major large alphabet compression scenarios, which are the classic lossless source coding problem, universal source coding and vector quantization. We show that in all of these cases, our suggested approach achieves a lower average codeword length than most commonly used methods.

All this together leads us to conclude that decomposing a large alphabet source into " as statistically independent as possible" components, followed by entropy encoding of each components separately, is both theoretically and practically beneficial.

Appendix

Proposition 1: Let $\underline{X} \sim \underline{p}$ be a random vector of an alphabet size m and joint probability distribution $\underline{p}$. The expected joint entropy of $\underline{X}$, when the expectation is over a uniform simplex of joint probability distributions $\underline{p}$ is

$$\mathbb{E}_{\underline{p}}\left\{H(\underline{X})\right\} = \frac{1}{\log_e 2}\left(\psi(m+1)-\psi(2)\right)$$

where ψ is the *digamma function*.

Proof: We first notice that a uniform distribution over a simplex of a size m is equivalent to a Direchlet distribution with parameters $\alpha_i = 1, i = 1, \dots, m$. The Direchlet distribution can be generated through normalized independent random variables from a Gamma distribution. This means that for statistically independent $Z_i \sim \Gamma(k_i = 1, \theta_i = 1), i = 1, \dots, m$ we have that

$$\frac{1}{\sum_{k=1}^m Z_k}(Z_1, \dots Z_m) \sim Dir\left(\alpha_1 = 1, \dots, \alpha_m = 1\right). \quad (33)$$

We are interested in the expected joint entropy of draws from (33),

$$\begin{aligned}\mathbb{E}_{\underline{p}}\left\{H(\underline{X})\right\} &= -\sum_{i=1}^m \mathbb{E}\left\{\frac{Z_i}{\sum_{k=1}^m Z_k}\log\frac{Z_i}{\sum_{k=1}^m Z_k}\right\}\\ &= -m\mathbb{E}\left\{\frac{Z_i}{\sum_{k=1}^m Z_k}\log\frac{Z_i}{\sum_{k=1}^m Z_k}\right\} \quad (34)\end{aligned}$$

It can be shown that for two independent Gamma distributed random variables $X_1 \sim \Gamma(\alpha_1, \theta)$ and $X_2 \sim \Gamma(\alpha_2, \theta)$, the ratio $\frac{X_1}{X_1+X_2}$ follows a Beta distribution with parameters (α_1, α_2). Let us denote $\tilde{Z}_i \triangleq \frac{Z_i}{\sum_{k=1}^m Z_k} = \frac{Z_i}{Z_i + \sum_{k\neq i} Z_k}$. Notice that $Z_i \sim \Gamma(1,1)$ and $\sum_{k\neq i} Z_i \sim \Gamma(m-1, 1)$ are mutually independent. Therefore,

$$f_{\tilde{Z}_i}(z) = Beta(1, m-1) = \frac{(1-z)^{(m-2)}}{B(1, m-1)}. \quad (35)$$

This means that

$$\begin{aligned}&\mathbb{E}\left\{\frac{Z_i}{\sum_{k=1}^m Z_k}\log\frac{Z_i}{\sum_{k=1}^m Z_k}\right\}\\ &= \mathbb{E}\left\{\tilde{Z}_i\log\tilde{Z}_i\right\} = \frac{1}{B(1,m-1)}\int_0^1 z\log(z)(1-z)^{(m-2)}dz\\ &= \frac{1}{B(1,m-1)}\frac{1}{\log_e(2)}\int_0^1 \log_e(z) z(1-z)^{(m-2)}dz\\ &= \frac{1}{m\log_e(2)}\mathbb{E}\left(\log_e(U)\right) \quad (36)\end{aligned}$$

where U follows a Beta distribution with parameters $(2, m-1)$. The expected natural logarithm of a Beta distributed random variable, $V \sim Beta(\alpha_1, \alpha_2)$, follows $\mathbb{E}\left(\log_e(V)\right) = \psi(\alpha_1) - \psi(\alpha_1+\alpha_2)$ where ψ is the *digamma function*. Putting this together with (34) and (36) we attain

$$\begin{aligned}\mathbb{E}_{\underline{p}}\left\{H(\underline{X})\right\} &= -m\mathbb{E}\left\{\frac{Z_i}{\sum_{k=1}^m Z_k}\log\frac{Z_i}{\sum_{k=1}^m Z_k}\right\}\\ &= \frac{1}{\log_e(2)}\left(\psi(m+1)-\psi(2)\right) \quad (37)\end{aligned}$$

■

Acknowledgment

This research was preformed while the first author was a doctoral candidate at Tel Aviv University, supported in part by a Returning Scientists grant from the Israeli Ministry of Science, and by Israeli Science Foundation grant 1487/12.

References

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

[2] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Commun. ACM*, vol. 30, no. 6, pp. 520–540, Jun. 1987.

[3] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Hoboken, NJ, USA: Wiley, 2012.

[4] L. D. Davisson, "Universal noiseless coding," *IEEE Trans. Inf. Theory*, vol. IT-19, no. 6, pp. 783–795, Nov. 1973.

[5] A. Painsky, S. Rosset, and M. Feder, "Generalized binary independent component analysis," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun./Jul. 2014, pp. 1326–1330.

[6] J. L. Massey, "Guessing and entropy," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun./Jul. 1994, p. 204.

[7] A. Moffat and A. Turpin, "On the implementation of minimum redundancy prefix codes," *IEEE Trans. Commun.*, vol. 45, no. 10, pp. 1200–1207, Oct. 1997.

[8] E.-H. Yang and Y. Jia, "Universal lossless coding of sources with large and unbounded alphabets," in *Numbers, Information and Complexity*. Norwell, MA, USA: Springer, 2000, pp. 421–442.

[9] A. Moffat, R. M. Neal, and I. H. Witten, "Arithmetic coding revisited," *ACM Trans. Inf. Syst.*, vol. 16, no. 3, pp. 256–294, 1998.

[10] J. Shtarkov, "Coding of discrete sources with unknown statistics," *Topics Inf. Theory*, vol. 23, pp. 559–574, 1977.

[11] A. Orlitsky and N. P. Santhanam, "Speaking of infinity," *IEEE Trans. Inf. Theory*, vol. 50, no. 10, pp. 2215–2230, Oct. 2004.

[12] W. Szpankowski and M. J. Weinberger, "Minimax pointwise redundancy for memoryless models over large alphabets," *IEEE Trans. Inf. Theory*, vol. 58, no. 7, pp. 4094–4104, Jul. 2012.

[13] A. Orlitsky, N. P. Santhanam, and J. Zhang, "Universal compression of memoryless sources over unknown alphabets," *IEEE Trans. Inf. Theory*, vol. 50, no. 7, pp. 1469–1481, Jul. 2004.

[14] I. H. Witten, A. Moffat, and T. C. Bell, *Managing Gigabytes: Compressing and Indexing Documents and Images*. San Mateo, CA, USA: Morgan Kaufmann, 1999.

[15] S. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inf. Theory*, vol. IT-28, no. 2, pp. 129–137, Mar. 1982.

[16] H. B. Barlow, T. P. Kaushal, and G. J. Mitchison, "Finding minimum entropy codes," *Neural Comput.*, vol. 1, no. 3, pp. 412–423, 1989.

[17] P. Comon, "Independent component analysis, A new concept?" *Signal Process.*, vol. 36, no. 3, pp. 287–314, 1994.

[18] A. Painsky, S. Rosset, and M. Feder, "Generalized independent component analysis over finite alphabets," *IEEE Trans. Inf. Theory*, vol. 62, no. 2, pp. 1038–1053, Feb. 2016.

[19] E. Arikan, "An inequality on guessing and its application to sequential decoding," *IEEE Trans. Inf. Theory*, vol. 42, no. 1, pp. 99–105, Jan. 1996.

[20] D. Malone and W. G. Sullivan, "Guesswork and entropy," *IEEE Trans. Inf. Theory*, vol. 50, no. 3, pp. 525–526, Mar. 2004.

[21] C. E. Pfister and W. G. Sullivan, "Rényi entropy, guesswork moments, and large deviations," *IEEE Trans. Inf. Theory*, vol. 50, no. 11, pp. 2794–2800, Nov. 2004.

[22] M. M. Christiansen and K. R. Duffy, "Guesswork, large deviations, and Shannon entropy," *IEEE Trans. Inf. Theory*, vol. 59, no. 2, pp. 796–802, Feb. 2013.

[23] I. Bairamov, A. Berred, and A. Stepanov, "Limit results for ordered uniform spacings," *Statist. Papers*, vol. 51, no. 1, pp. 227–240, 2010.

[24] R. M. Young, "Euler's constant," *Math. Gazette*, vol. 75, no. 472, pp. 187–190, 1991.

[25] D. G. e Silva, R. Attux, E. Z. Nadalin, L. T. Duarte, and R. Suyama, "An immune-inspired information-theoretic approach to the problem of ICA over a Galois field," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Oct. 2011, pp. 618–622.

[26] B. New, C. Pallier, M. Brysbaert, and L. Ferrand, "*Lexique 2*: A new French lexical database," *Behavior Res. Methods, Instrum., Comput.*, vol. 36, no. 3, pp. 516–524, 2004.

[27] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Statist. Probab.*, vol. 1. 1967, pp. 281–297.

[28] P. A. Chou, T. Lookabaugh, and R. M. Gray, "Entropy-constrained vector quantization," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 37, no. 1, pp. 31–42, Jan. 1989.

[29] A. Gersho, "Asymptotically optimal block quantization," *IEEE Trans. Inf. Theory*, vol. IT-25, no. 4, pp. 373–380, Jul. 1979.

[30] R. Zamir, *Lattice Coding for Signals and Networks: A Structured Coding Approach to Quantization, Modulation, and Multiuser Information Theory*. Cambridge, U.K.: Cambridge Univ. Press, 2014.

Amichai Painsky received his B.Sc. in Electrical Engineering from Tel Aviv University (2007), his M.Eng. in Electrical Engineering from Princeton University (2009) and his Ph.D. in Statistics from the School of Mathematical Sciences in Tel Aviv University (2016). He is currently a Postdoctoral Fellow, co-affiliated with the Israeli Center of Research Excellence in Algorithms, located at the Hebrew University of Jerusalem, and with the Signal, Information and Algorithms Laboratory at Massachusetts Institute of Technology (MIT). His research interests include Data Mining, Machine Learning, Statistical Learning and their connection to Information Theory.

Saharon Rosset is an Associate Professor in the department of Statistics and Operations Research at Tel Aviv University. His research interests are in Computational Biology and Statistical Genetics, Data Mining and Statistical Learning. Prior to his tenure at Tel Aviv, he received his PhD from Stanford University in 2003 and spent four years as a Research Staff Member at IBM Research in New York. He is a five-time winner of major data mining competitions, including KDD Cup (four times) and INFORMS Data Mining Challenge, and two time winner of the best paper award at KDD (ACM SIGKDD International Conference on Knowledge Discovery and Data Mining).

Meir Feder (S'81–M'87–SM'93–F'99) received the B.Sc and M.Sc degrees from Tel-Aviv University, Israel and the Sc.D degree from the Massachusetts Institute of Technology (MIT) Cambridge, and the Woods Hole Oceanographic Institution, Woods Hole, MA, all in electrical engineering in 1980, 1984 and 1987, respectively.

After being a research associate and lecturer in MIT he joined the Department of Electrical Engineering - Systems, School of Electrical Engineering, Tel-Aviv University, where he is now a Professor and the incumbent of the Information Theory Chair. He had visiting appointments at the Woods Hole Oceanographic Institution, Scripps Institute, Bell laboratories and has been a visiting professor at MIT. He is also extensively involved in the hightech industry as an entrepreneur and angel investor. He co-founded several companies including Peach Networks, a developer of a server-based interactive TV solution which was acquired by Microsoft, and Amimon a provider of ASIC's for wireless high-definition A/V connectivity.

Prof. Feder is a co-recipient of the 1993 IEEE Information Theory Best Paper Award. He also received the 1978 "creative thinking" award of the Israeli Defense Forces, the 1994 Tel-Aviv University prize for Excellent Young Scientists, the 1995 Research Prize of the Israeli Electronic Industry, and the research prize in applied electronics of the Ex-Serviceman Association, London, awarded by Ben-Gurion University.