Entropy Adaptive On-Line Compression

by

Shlomi Dolev, Sergey Frenkel and Marina Kopeetsky

Technical Report #14-04

July 2014

Entropy Adaptive On-Line Compression

(Technical Report)

Shlomi Dolev* Sergey Frenkel† Marina Kopeetsky‡

July 3, 2014

Abstract

Self-organization is based on adaptivity. Adaptivity should start with the very basic fundamental communication tasks such as encoding the information to be transmitted or stored. Obviously, the less signal transmitted the less energy in transmission used. In this paper we present a novel on-line and entropy adaptive compression schemes for streaming unbounded length inputs. The scheme extends the window dictionary Lempel-Ziv compression and is adaptive and is tailored to compress on-line non entropy stationary inputs. Specifically, the window dictionary size is changed in an adaptive manner to fit the current best compression rate for the input. On-line Entropy Adaptive Compression scheme (EAC), that is introduced and analyzed in this paper, examines all possible sliding window sizes over the next input portion to choose the optimal window size for this portion, a size that implies the best compression ratio. The size found is then used in the actual compression of this portion. We suggest an adaptive encoding scheme, which optimizes the parameters block by block, and base the compression performance on the optimality proof of *LZ77* when applied to blocks [25]. The EAC scheme was tested over files of different types (docx, ppt, jpeg, xls) and over synthesized files that were generated as segments of homogeneous Markov Chains. Our experiments demonstrate that the EAC scheme typically provides a higher compression ratio than *LZ77* does, when examined in the scope of on-line per-block compression of transmitted (or compressed) files.

1 Introduction

One of the most challenging task of communicating entities in distributed systems and computer communication networks is the way information is compressed. As the information is better compressed the bandwidth and energy used for communication are reduced and the performance of the

*Department of Computer Science, Ben-Gurion University, Beer-Sheva, 84105, Israel. Email: hd@cs.bgu.ac.il. Partially supported by a Russian Israeli grant from the Israeli Ministry of Science and Technology and the Russian Foundation for Basic Research, the Rita Altura Trust Chair in Computer Sciences, the Lynne and William Frankel Center for Computer Sciences, Israel Science Foundation (grant number 428/11), Cabarnit Cyber Security MAGNET Consortium, Grant from the Institute for Future Defense Technologies Research named for the Medvedi of the Technion, MAFAT, and Israeli Internet Association

†Department of Computer Science, Ben-Gurion University, Beer-Sheva, 84105, Israel. Email:dolev@cs.bgu.ac.il. Partially supported by Deutsche Telekom, Rita Altura Trust Chair in Computer Sciences, Lynne and William Frankel Center for Computer Sciences, Israel Science Foundation (grant number 428/11) and Cabarnit Cyber Security MAGNET Consortium.

‡Department of Communication Systems Engineering, Ben-Gurion University, Beer-Sheva, 84105, Israel. Email: zvilo@bgu.ac.il.

distributed system becomes more efficient. We seek for adaptive compression schemes that provide the best compression for different and non uniform data flows. The sole traditional requirement for the optimality of lossless data compression schemes is the stationary ergodic nature of the information source. Nevertheless, due to the wide deployment of multimedia networks, heterogeneous ad-hoc network, and the wide range of communication tasks, the efficient compression techniques of the dynamic (non-stationary) sources are of a special interest now-days. In particular, dynamic sources that are characterized by non-stationary probability distribution and non constant entropy, are the typical sources that transmit on-line multimedia (voice, video) traffic. To the best of our knowledge, we provide the first practical and efficient on-line adaptive scheme that tracks the variable entropy rate of the source and provides optimal compression of fixed-size data blocks on-line without performing computationally and time expensive preprocessing.

Related work. A universal coding scheme for sequential data compression was introduced firstly in [26] and [27]. Lempel and Ziv introduced compression algorithm defining a rule for parsing strings of symbols from a finite alphabet into substrings, or words of bounded length, and a coding scheme which maps these substrings sequentially into uniquely decipherable codewords of fixed length over the same alphabet. It has been demonstrated that as the sliding window size tends to infinity, the compression ratio approaches the source entropy.

Two algorithms, based on incremental parsing, namely $LZ77$ and $LZ78$, have been introduced and analyzed. The main difference between $LZ77$ and $LZ78$ algorithms is that $LZ77$ algorithm is based on a fixed dictionary size (or, equivalently fixed memory size), while the dictionary size of $LZ78$ algorithm may grow infinitely. The sliding window Lempel-Ziv algorithm $LZ77$ and its asymptotic optimality were analyzed in [10] and [22]. As for non-asymptotic coding for finite data streams, some theorems were derived in [9]. The performance of the $LZ78$ and $LZ77$ algorithms is studied without any initial assumption on the input. It is demonstrated, that the standard definition of optimal compression does not take into account the performance of compression algorithms when the input is a low entropy string. Moreover, there exist families of low entropy strings which are not compressed optimally.

However, there is a great interest in on-line data compression when, unlike the described above off-line methods, the only available information is the currently processed block of data. Ziv has proven in [27] that the $LZ77$ universal compression of N-blocks is essentially optimal for finite N-blocks. Hence, the asymptotically optimal data compression algorithm $LZ77$ is also optimal when the data block is of finite length.

On-line compression scheme has been proposed in [6]. The proposed algorithm makes on-line decisions whether to compress a file or send it in the original non compressed form. The file compression is only performed when it can improve the file transmission time. A Lempel-Ziv-Welch LZW lossless compression algorithm (that belongs to the family of the LZ algorithms) was adopted. The simulation results in [6] reveal that compression may lead to a longer overall delay under light traffic loads, while it can significantly reduce the delay under heavy traffic loads and increase maximum throughput.

Variable-length coding combined with the Lempel-Ziv technique, is proposed in [12] for reducing the size of large messages. The new practical Neural Markovian Predictive Compression ($NMPC$) algorithm for obtaining lossless on-line compression has been designed and tested in [18]. $NMPC$ algorithm is based on the Bayesian Neural Networks (BNN) and Hidden Markov Models (HMM). The experimental results demonstrate that $NMPC$ algorithm performs best (even better than the dictionary based Lempel-Ziv family algorithms) when the input includes predictable statistical

patterns that can be learned by BNN and HMM. However, this approach requires a significant amount of preprocessing.

A universal variable-length lossless compression algorithm based on error correcting fountain codes has been introduced in [5]. The proposed method is based on the Belief Propagation algorithm in conjunction with the iterative doping algorithm and the inverse Burrows-Wheeler block sorting transform. It is demonstrated that the compression scheme is effective for the non stationary sources. Nevertheless, unlike our EAC scheme, the method of [5] is probabilistic and explicitly computes the source statistics. We also note the similar approaches suggested in [7] and [3].

Our contribution. The scope of this paper is the on-line data compression by sequential application of $LZ77$ algorithm to individual finite-length of non-overlapping B-blocks of data, where B is the blocks length. We present the EAC scheme that dynamically tracks the sliding window size n_w without explicit measurement of the source entropy. The optimal or near optimal n_w, is computed based on the analysis of the buffered look ahead data, that is permanently generated by the source. The EAC scheme computes the optimal window size on-line given a predefined communication latency, or size of buffered data, which is facilitated by a look ahead buffer in which the very next portion of the read data is accumulated. Compared with the recently proposed schemes, the EAC scheme has the following advantages.

Optimality of the compression ratio. The currently best sliding window size for each individual N-block, that implies the minimal codeword length, is computed on-line and applied by the EAC scheme. Comparing with [16], in order to achieve the optimal compression, our scheme demands that the length of the stationary component s_i, generated by the information source, is not shorter than the optimal window size n_{w_i} that yields the entropy H_i of s_i. In order to compute the optimal sliding window size n_w, that satisfies maximal compression ratio, the value of n_w grows exponentially, each is twice the size of the preceding. Hence, there are no redundant bits in the binary representation of the encoded phrase (codeword).

Robustness for the non-stationary sources. The sliding window size is changed in an adaptive manner to always fit the file structure, while the current entropy of the source may not been explicitly measured. The EAC scheme effectively tracks any changes of data generated by a random (possibly non stationary) source.

Low computational cost combined with fast adaptivity rate. Unlike the dynamic Huffman coding and the $NMPC$ schemes ([11], [21] and [18]), the EAC scheme does not need significant preprocessing. Moreover, the compression performed by the EAC scheme achieves the maximal compression ratio since we adaptively track the window size, and choose the best to compress the current buffered generated data. Unlike [6], the EAC scheme is performed permanently without significant overhead of traffic measurement and mathematical computations that are based on a queuing model and the prediction of the overall network delay.

Efficiency in case of fixed memory size. The memory size, kept at the encoder and the decoder sides, is fixed. As a result, the EAC scheme is a practical and efficient compression scheme that can be implemented in a computer environment with a restricted memory. For example, dynamic Huffman coding ([11] and [21]) requires exponentially large dictionary, in order to approach the optimal compression possible for a given entropy.

The simulation results, presented in Section 4 demonstrate that the EAC scheme can perform in many cases better and achieves higher compression ratio on-line, comparing with the standard $LZ77$ compression scheme. Hence, the EAC scheme may be plugged in other window based lossless compression schemes (e.g., [6, 12, 18, 24]) in order to increase the compression ratio and improve

the overall performance of a given compression algorithm.

Paper organization. Section 2 presents the settings describing the $LZ77$ dictionary based compression scheme, which is the base for the EAC scheme. The Entropy Adaptive Compression EAC scheme is introduces and analyzed in Section 3. Discussion and analysis of the experimental results appear in Section 4. At last, conclusions can be found in Section 5.

2 Preliminaries

Lossless data compression scheme, based on the dictionary method, is the basic for our EAC scheme. The EAC scheme is based on the sliding window Lempel-Ziv $LZ77$ algorithm that was proposed in [26] and further analyzed in [9, 10, 22, 23]. In the $LZ77$ scheme the dictionary consists of strings from the sliding window into recently generated text. Let $(X_k^{k=+\infty}{}_{k=-\infty})$ be a stationary random process with entropy H. $(X_k^{k=+\infty}{}_{k=-\infty})$ generates random strings over finite alphabet A. Without loss of generality let us assume a binary case when $A = \{0, 1\}$.

The encoding of the sequence $\{X_k\}_{k=1}^N$, where N is a large integer, is performed as follows. Let n_w be an integer parameter, called the *sliding window size*. The values of n_w are restricted to powers of two values (as indexes in the window are always represented by binary values). The first n_w symbols of $X_1^{n_w}$, called *training sequence*, are encoded by the binary encoding algorithm [22] with no attempt for compression (see below). The binary encoding scheme (or mapping) e unambiguously encodes an integer L into a binary string such that for any distinct integers $L_1 \neq L_2$ $e(L_1)$ is not a prefix of $e(L_2)$. Thus the code is uniquely decipherable. See [26] for detailed description of the $LZ77$ encoding-decoding process. The major benefit of the $LZ77$ compression algorithm is that it yields the ultimate compression in the asymptotic case. Namely, it compresses a data source according to the maximal compression ratio for the given entropy, if the sliding window size n_w and the length of the sequence $\{X_k\}_{k=1}^N$ both tend to infinity [22]. We consider the non-asymptotic case, which is characterized by the restricted memory size at the encoder and the decoder sides. Since the lengths of the training sequence and the sliding window size are both fixed, the optimal ultimate compression (at the entropy rate) cannot be achieved. Typically, authors consider that the mathematical models of the $LZ77$ algorithm are parametrized by the following parameters: the size of sliding window and the maximal length of phrases [14, 22], entropy of underlying process, length of stationary segments with their entropy estimations in the case of non-stationary process [16].

The Asymptotic Equipartition Property theorem, AET, [20], is commonly used. AET is a consequence of the law of large numbers and the ergodic theory. It states the following: consider the series of ergodic sequences that may be generated by a random source. Then, asymptotically, almost all sample paths of the stationary ergodic process have the same entropy rate. This implies the existence of "typical" sequences. However, since the rate of convergence towards the AET is not uniform over the various ergodic stationary sources and may be very slow, n_w might be very large. Furthermore, it is not known how the LZ family of algorithms perform in the case of small sliding window sizes. In fact, as the sliding window is considered as a "training sequence" for the information source, it should be large enough in comparison with the compressed data. Hence, it is interesting to perform experiments with real-life files. Note, that originally the $LZ77$ algorithm provides estimation of the compression ratio as a result of an entropy estimation, which is, a reciprocal of the compression ratio defined as $CR = L_0/L_{LZ}$, where L_0 and L_{LZ} are the lengths of the original and compressed file, respectively.

The non-asymptotic coding and limitations on the sliding window size were derived firstly in [9, 10] for lossless data compression algorithms with fixed statistical side information while the training sequence is not large enough. The converse and coding theorems, that state the relation between entropy of the source and corresponding sliding window size (training sequence length), were derived. It was demonstrated (Theorem 3.1) that if the sliding window size n_w is smaller than $2^{l(H-\epsilon)}$, where H is the entropy of the source, l is a large enough codeword length, and ϵ is arbitrarily small, then there exists a number of incompressible sources. Nevertheless, a compression that is close to the optimal for a given entropy is possible if $n_w \geq 2^{l(H+\epsilon)}$ for sufficiently large l and arbitrarily small ϵ. As a matter of fact, the stationary sources that generate strings with a constant entropy, were treated in [9, 10]. Nevertheless, in these papers the sliding window size is fixed and determined by the constant entropy of the stochastic source.

In our scope, information source is not a stationary source, therefore the entropy of its input is, in essence, a function of time. It is known that if the entropy is fixed, then the original sliding window Lempel-Ziv $LZ77$ algorithm converges to it [22]. The EAC scheme is intended to bound the difference between the current window size we work with, compared to the optimal one, had we known the "instantaneous" entropy rate. The following complexity considerations regarding the sliding window size should be taken into account. On the one hand, smaller window size leads to a more efficient compression achieved by the shorter codeword length. On the other hand, as n_w becomes larger, the longer are the strings that may be compressed efficiently, and the number of codewords is lower. Nevertheless, as n_w becomes larger, the number of incompressible phrases (that are shorter than $\log n_w$) becomes larger.

3 Description of the EAC scheme.

Let B denotes the number of look ahead (LA) buffered bits. The optimal window size n_w is computed for encoding (compression) of any portion of B bits of the whole file. B determines the latency, and within the compression and transmission of B bits n_w is not changed. The computation of the optimal window size n_w is based on the analysis of the dictionary that consists from the previously sent data, and on comparing the compression ratio in the consequently decreasing windows. Since the encoder E and the decoder D are not synchronized, E has to update D with the value of n_w, optimized for the compression of the current portion of the (look ahead) buffered B bits.

The trade-off between the possible values of B should be taken into consideration. On the one hand, small B leads to a small transmission delay. Nevertheless, the value of n_w is not stable as we compute it over very small part of the entire file. On the other hand, large B implies large transmission delay. Yet, the computed optimal n_w is more stable.

The *EAC* scheme with *B*-bit delay. The detailed description of the scheme is presented in Figure 1. Let the information source generate a random (may be non-stationary) finite length string. Let the initial sliding window size $N_0 = n_{w_0}$ be set during the training stage as the maximum possible window size, based on a certain training sequence by applying the original $LZ77$ algorithm. The first n_{w_0} bits of the initial window from the input are sent from E to D (by agreed upon efficient algorithm (e.g., $LZ77$)), (lines 2-8). Upon reception of B bits, generated by the source, the encoder E computes the optimal window size N. The initial window N_0 is used as a pyramid base for testing all possible smaller windows. The test stage for the current portion of size B bits is performed

by trying all windows of sizes $N_0/2^i$ for every $0, \ldots, \log N_0$. The EAC algorithm starts using a dictionary of $N_0/2^i$ bits from the previously received and compressed B bits, and then shift the window which is also of size $N_0/2^i$ until the algorithm is done with the current portion of B bits (lines 8-10, 14-29). The total length of each encoded phrase is composed from the comma free binary encoding of its length L_i (denoted by $e(L_i)$), and the binary encoding of the corresponding index m_i [9]. The total length of the compressed string determines the redundancy that has been removed from the original uncompressed B- bits string. The average compression ratio in the $i-th$ window is $CR_i = \frac{B}{\sum e(L)+\log n_{w_i}+\log(i+1)} = \frac{B}{\sum e(L)+i+\log(i+1)}$, determining the average compression quality in each $i-th$ window. The window size $N = n_{w_i}$, that satisfies the shortest length of the compressed string (and corresponding maximal compression ratio), is determined as the current optimal size (lines 30-33). The current portion of B bits is compressed using the optimal N and sent to D (line 10). If the window size has been updated, its new value is inserted to the transmitted string.

The strictly on-line implementation of the EAC scheme with the negligible loss in compression quality is also possible. In the case of the Hard Real Time system, when there is no access to the previous data (history) at the encoder E or the decoder D sides, the EAC scheme may be implemented strictly on-line, by the consequent exponential increase of the sliding window. Nevertheless, a certain loss in compression quality cannot be avoided. Assume that the information source is a dynamic source [16] that generates sequential stationary strings, each characterized by a distinct entropy, and let s_1 and s_2 be two such strings. Each string $s_i,\ i = 1,\ 2$ is characterized by its constant entropy H_i and corresponding optimal window size n_{w_i}. It should be reminded, that the source entropy is unknown. In the following example we explain the change in the window size for two interesting cases.

Case 1: $H_2 > H_1$. In this case the low entropy string s_1 changes to the high entropy string s_2, and the sliding window size should be increased by multiplying it by 2^k for a certain integer $k > 0$. If the encoder and the decoder do not maintain the last B bits transmitted, but only the last n_{w_1} bits, then E cannot encode phrases immediately in the optimal n_{w_2} since E and D cannot return to the previous bits of the input, necessarily for the decoding of the received strings using n_w. In such a way a loss in compression quality occurs. E continues (non optimally) encoding using the previous small window n_{w_1}. Nevertheless, in order to increase the encoding quality, E doubles window size in a slow start fashion, starting from n_{w_1}, as soon as the decoder D receives the number of bits, required for the decoding using the larger window. Therefore, the consequent window sizes, used by E and D simultaneously, are $2n_{w_1}$ .. $2^k n_{w_1}$.

Case 2: $H_2 < H_1$. In this case the size n_{w_2} of the newly computed optimal window is smaller than the current window size n_{w_1}, namely $n_{w_2} = 2^{-s} n_{w_1}$ for a certain $s > 1$. In this case the decoder D has the required number of bits in its history, and the new optimal window n_{w_2} is contained in the previous (larger) window n_{w_1}. Hence, there is no loss in compression quality in the encoding/decoding procedure.

In case of a strictly on-line implementation of the EAC scheme, the minimal loss in the compression quality occurs when the source entropy is increased from H_1 to H_2, $H_1 \leq H_2$ (Case 1). The loss in bits is estimated as $(1 - H_2) \cdot l$. Here the term $1 - H_2$ is the loss of optimality in compression for a single bit, and l is the number of bits in the non-optimally compressed sample. Nevertheless, there is no loss in the compression quality in Case 2 when the source entropy decreases.

```
1: EAC scheme for encoder E
2: Loop over whole file for each portion of B bits
3: int B
4: /* number of look ahead buffered bits respecting the maximal allowed latency */
5: int N_0 initial window size
6: int N_prev window size optimal for compression of the previous portion of B bits
7: /* Bootstrap stage – establishing the first dictionary */
8: Compress the first N_0 bits by agreed upon efficient algorithm (e.g., LZ77) and send to
   decoder D
9: Upon the arrival of the next B bits of the (streaming) file
10: TestCompress(B, N_0, Output)
11: Send Output to decoder D
12:
13:
14: Procedure TestCompress(LA, PB, Output)
15: /* Procedure TestCompress: search for the optimal window size N ≤ PB
16: for the portion of LA bits from input/*
17: Input:
18: int LA length in bits of InputString for compression
19: N_0 = PB initial window size (pyramid base)
20: Perform LZ77 compression of LA bits using the last N_0 bits of previous LA
21: as the dictionary
22: CompressedString = Output
23: /* CompressedString – LA bits, compressed in optimal window */
24: Compute A – length of CompressedString in bits
25: for int i = 1 .. log PB
26:     Perform LZ77 compression of LA bits using the last PB/2^i bits of previous LA
27:     as the dictionary and
28:     Compute length L_i of string CompressedString_i
29:     using window of size n_{w_i} = PB/2^i bits
30:     N = PB
31:     /* N optimal window size for LA bits */
32:     if L_i < A
33:         Set A = L_i, N = n_{w_i}, CompressedString = CompressedString_i
34: if N = N_prev
35: Output = (N, CompressedString)
```

Figure 1: Entropy Adaptive Compression Scheme.

4 Analysis and Experimental Results

Experimental comparison of *LZ77* vs. *EAC*. The EAC scheme was tested with different real-life files of different types (docx, ppt, jpeg, xls), and some artificial ones generated as segments of homogeneous Markov Chains. As the maximal possible $LZ77$ sliding window size N_0 must not be larger than the size of a compressed file, we use the pyramid base N_0 for each B bits segment equal to B ($N_0 = B$). Figures 2, 5, and 6 demonstrate that the EAC scheme may provide a compression ratio higher, compared to the $LZ77$ algorithm, for the on-line per-block compression of the transmitted files.

Let us summarize the current principal theoretical results regarding the compression of a file by the $LZ77$ algorithm, applied in the case of sequential blocks of the entire file.

Ziv showed in [25] that the $LZ77$ algorithm is asymptotically optimal when applied on-line

to consecutive strings of length B of M blocks, as M tends to infinity. It means that if the $LZ77$ algorithm is applied to each consecutive block, the compression at entropy rate is achieved asymptotically if the number of blocks M is very large. As the compression ratio of encoding of any ergodic random sequence is lower bounded by its entropy, we assume the optimal compression, compared to the methods that do not use any a priory information about probabilistic distribution of the sequences. It is essential that the influence of the sliding window size on the compression ratio is not considered in [25], as the issue studied is whether it is possible to estimate the entropy exactly using only individual B-blocks. It means that the sliding window size of the $LZ77$-algorithm in [25] is bounded by the value of B.

In fact, [10] also considers some issues of $LZ77$ encoding in the cases of finite sequences, with the sliding window n_w of bounded size. The assumptions of [10] and [25] are similar in a sense that both of them deal with the fixed case, as a finite sequence of size N in [25] means that the sliding window, used by the $LZ77$ algorithm in [25], cannot be larger than B. Therefore, these theoretical results demonstrate that our practical scheme can provide the optimal compression like $LZ77$ algorithm, if $B \cdot M$ is very large, where M denotes the number of B-bit blocks, and B is larger than some threshold value [25]. More precisely, it means that at least for some sets of sequences, an essentially optimal algorithm allows for a rapid convergence to the asymptotic complexity, if the length of the string is exponentially (more than $B^{1-\epsilon}$) larger than a length for which no effective compression is possible by any lossless compression algorithm. Here $\epsilon > 0$ is an arbitrary small number, such that $\epsilon << (1-k)\log A$ (i.e., the compression ratio achieved for each B-block of a infinite sequence X is smaller than reciprocal of its entropy $H(X)$). It means, that for any given file, characterized by its entropy H and by a parameter $0 < k < 1$ such that $H < k \log A$, there exists a threshold value of file size (namely, $B' \leq B^{1-\epsilon}$ for less of which it is impossible to compress it by any data compression algorithm).

Generally speaking, an empirical entropy $H_B(N)$, in the theoretical framework of [10], determines the compression ratio of a finite random discrete sequence for non overlapping B-bits blocks that appear in a sequence of the length $B \cdot M$ bits [25], as this quantity is similar to the classical definition of the empirical entropy of B-blocks in an individual sequence of length $B \cdot M$ [25].

The following parameters that impact the compression ratio are considered in our experiments: estimation of empirical entropy for the investigated file, values of block size B, and pyramid base (maximal possible sliding window size N). Note, that the notion of "compression ratio" in [10] means the smallest number of bits per letter that can be asymptotically achieved by any B- bits block data-compression scheme for a random sequence X, generated by a random information source. Let us consider that X is characterized by a stationary probability measure P. Then, according to [10], it is possible to represent the mathematical expectation of the compression ratio of the compressed file by averaging over sliding windows as

$$CR = \frac{L}{\sum_{z \in A^l} Prob(X_1^l = z) L(z/x')},$$

where $x' = A^{n_w}$, $z = A^l$ are the sets of phrases among whole elements of original file of the fixed size (modeled as a string X_1^l) and the sliding window of size n_w over alphabet, L is the length of $LZ77$ code of the string z. Note that below we consider the compression ratio as $CR = L_0/L_{LZ}$, where L_0 is the length of the original file, L_{LZ} is the size of the compressed file. It can be easily seen that in these terms the compression ratio CR is equal to

$$CR = \frac{L_0}{B\left(\sum_{j=1 \ldots \frac{L_0}{B}} 1/CR_j\right)},$$

where B is the size of the portion of bits to be compressed (Section 3), which we consider as a power two, $j = 1..\ L_0/B$.

Let us use the estimation of the compression ratio for each of B-block from [22]. Since our "pyramid based" EAC algorithm (Section 3) computes the optimal window size for each B portion by a sequential decrement of the initial maximal possible n_w value, the enhancement in the compression quality can be achieved by the appropriate computation of the window size, that is optimal for the current portion of B bits. In order to fairly compare the EAC and $LZ77$ algorithms, the fixed window size n_w, used by the $LZ77$ algorithm, is equal to the maximal window size (pyramid base), applied in our scheme. Figures 2 - 6 demonstrate the impact of the different mentioned above parameters on the compression ratio of the EAC scheme (compared with the $LZ77$ algorithm).

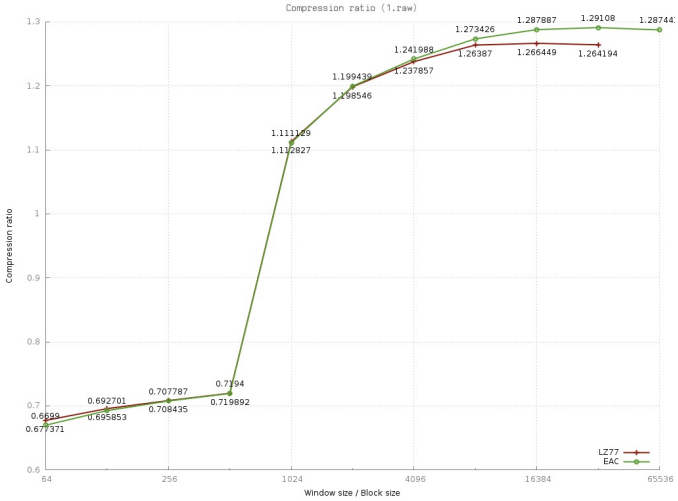

Figure 2: CR vs. window/block size. File size is 10Mbit.

As it was mentioned above, the analysis must take into account some entropy-like estimation of the information sources. The model of the information source is generated correspondingly to the sequences/files, computed by theoretic or empirical probabilistic measures of the sources. As the empirical measures are rather sophisticated, we may estimate information property of the files by the approximate formula

$$H_{n_w,k} = \left[\frac{1}{k}\sum_{i=1}^{k}\frac{L_i^{n_w}}{\log n_w}\right]^{-1}$$

for empirical entropy $H_{n_w,k}$ for a B-block, where k is the number of matches for a given B-block, n_w is the sliding window size, chosen for a given B-block by the EAC algorithm, i is the current position in the block [25]. That is the quantity $\frac{log\ n_w}{L_i^{n_w}}$ can be used as an entropy estimator.

As the various lengths of the longest matches (Figure 3) may vary in hundred times depending on window /block sizes, the average value of various match-lengths $L_i^{n_w}$, taken at different positions i, would be more reasonable. That is we will use $E(L)/E(\log n_w)$, where $E(L)$ is the average longest match over all blocks, and $E(\log(n_w))$ is the average size of the sliding window. The Figures 2, 5 and 6 demonstrate that the larger file size is, the closer the compression ratio of the EAC and the $LZ77$ algorithms.

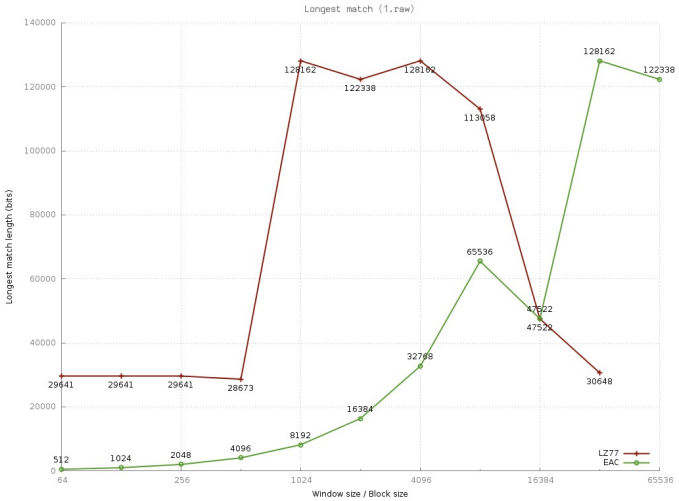

Figure 3: Average longest match length of EAC and $LZ77$ vs window/block size for the file of Figure 2.

This is the logical consequence of the expounded above asymptotic property theorem for LZ compression for B-block sequences [25]. A small improvement ($CR = 1.29$ vs $CR = 1.26$, Figure 2) of the EAC scheme for rather large values of blocks is a result of increasing of the lengths of the longest matches (Figure 3),that leads decreasing of empirical entropy, and, correspondingly, increasing the compassion ratio [23]. Besides, the EAC scheme can provide higher compression ratio for rather short sequences (Figures 5 and 6), as the asymptotic properties of theorem [25] are still not satisfied for such file sizes by using of LZ77.

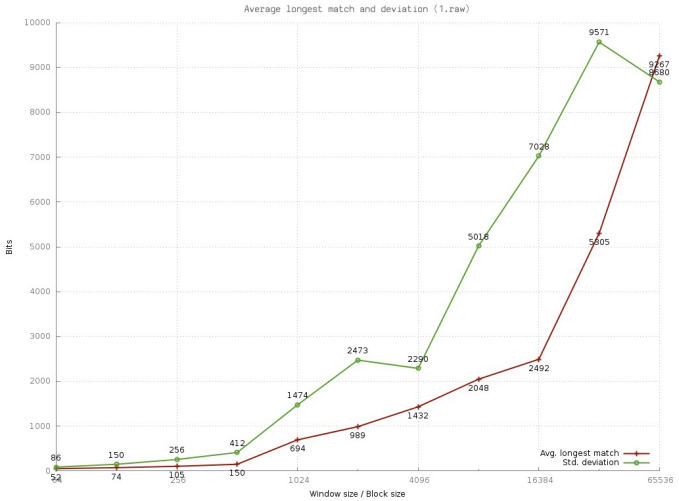

Figure 4: Average and variance of longest matches length for the file of Figure 2.

5 Conclusion

The technique of $LZ77$ may be effective if the statistics of the dictionary is proper for the rest of the encoded sequence (which will asymptotically dominate). Since, as a rule, optimal choice of the n_w is less essential, this requirement cannot be satisfied for the non-stationary process case. In contrast, the EAC scheme, suggested in this paper, performs the $LZ77$ window based encoding only for a small B portion of the file. Hence, the non-stationary nature of the data affects the compression algorithm, and taking into account of this aspect is also a factor of the increase of the compression ratio of the EAC scheme. As a result, the EAC scheme can provide a high compression ratio (compared with the $LZ77$ algorithm), especially for rather short sequences (Figures 5 and 6).

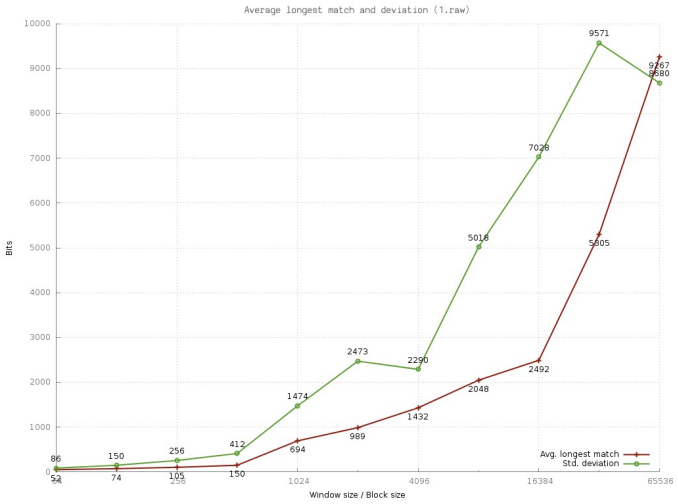

Figure 5: CR vs window/block size. File size is 0.93Mb.

Using a large window size n_w, the estimators are more likely to capture the longer-term trends in the data, although a large window size will give estimates with high variance (see Figure 4). However, based on [22], the window size is a factor of the $LZ77$ overhead. Nevertheless, the larger is the fraction of the compressed sequence which was already analyzed (that is used as a dictionary), the larger will be the probability to find the longest match in the rest of the file. This probability can be increased by the optimization the coding for the dictionary part (by testing the sliding window sizes in order to determine its optimal value), while the rest of the file is encoded based on the dictionary. This technique may be effective if the statistics of the dictionary is proper for the rest of encoded sequence (which will asymptotically dominate). This requirement cannot be satisfied for the non-stationary process. In opposite, the EAC scheme performs the $LZ77$ window based encoding only for a small B portion of the file.

Actually, the effectiveness of the standard $LZ77$ algorithm may be evaluated by the probability that the substrings from the dictionary are contained in the current portion of B bits. Consider a binary file, generated by an information source the file be the binary one, generated by the information source with a certain known probabilistic characteristics, say, a Bernoulli trial. Let A and B are two information sources such as A generates a *dictionary* and B generates a B-portion of the Look Ahead LA bits. Then, the number of bits per character, wasted to encode the sequence emitted by B with the optimal encoding for A, can be estimated by the relative entropy

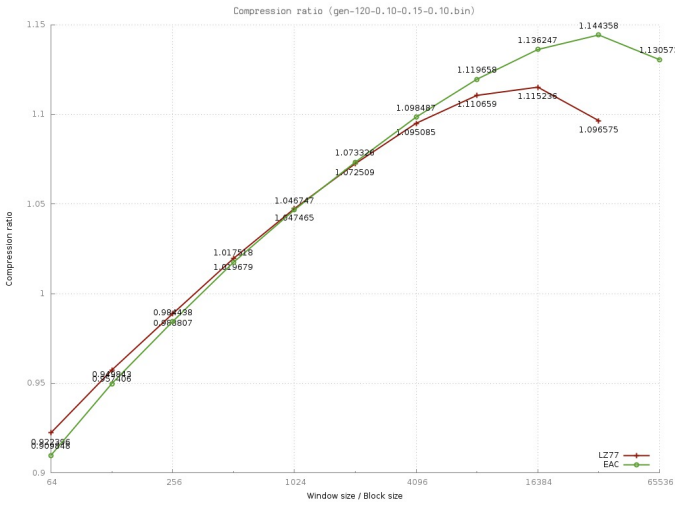

Figure 6: CR vs. window/block size. File size is 32K.

per character of A and B as suggested in [15].

Acknowledgment. We thanks Asaf Cohen for useful remarks and discussions.
We thanks Dmitry Zbarsky for implementing and testing our Entropy Adaptive Compression scheme.

References

[1] A. Baronchelli, E. Caglioti and V. Loreto, "Measuring complexity with zippers", in *ArxiVe 2006*.

[2] T. Bell, D. Kulp, "Longest-match String Searching for Lempel-Ziv Compression", *Software-Practice and Experience Journal*, vol. 23, no. 7, pp. 757-771, 1993.

[3] A. Beimel, S. Dolev, N. Singer, "RT Oblivious Erasure Correcting", *IEEE/ACM Trans. Netw.*, vol. 15, no. 6, pp. 1321-1332, 2007.

[4] P. E. Bender, J. K. Wolf, "New Asymptotic Bounds and Improvements on the Lempel-Ziv Data Compression Algorithm", *IEEE Transactions on Information Theory*, vol. 37, no. 3, pp. 721-729, 1991.

[5] G. Caire, S. Shamai, A. Shokrollahi , S. Verdu, "Universal variable-length data compression of binary sources using fountain codes", *IEEE Information Theory Workshop*, 2004.

[6] X. Deng, Y. Yang, "On-Line Adaptive Compression in Delay Sensitive Wireless Sensor Networks", *IEEE Transactions on Computers*, vol.61, issue 10, pp. 1429-1442, 2012.

[7] S. Dolev, B. Fitingof, A. A. Melkman, O. Tubman, *Smooth and Adaptive Forward Erasure Correcting*, Computer Networks Journal, vol. 36, no. 2/3, pp. 343-355, 2001.

[8] D. Hankerson, G. A. Harris, P. D. Johnson, *Introduction to Information theory and Data Compression*, Chapman and Hall/CRC, second edition, 2003.

[9] Y. Hershkovits, J. Ziv, "On Fixed-database Universal Data Compression with Limited Memory", *IEEE Transactions on Information Theory*, vol. 43, no. 6, pp. 1966-1976, 1997.

[10] Y. Hershkovits, J. Ziv, "On Sliding-Window Universal Data Compression with Limited Memory", *IEEE Transactions on Information Theory*, vol. 44, no. 1, pp. 66-78, 1998.

[11] D. E. Knuth, "Dynamic Huffman Coding", *Journal of Algorithms*, no.6, pp. 163-180, 1985.

[12] K. R. Kolhe, P. R. Devale, P. Shrivastava, "High performance Multimedia Data Compression through Improved Dictionary", *International Journal of Computer Applications*, vol. 10, no. 1, 2010.

[13] S. Rao Kosaraju, G. Manzini, "Compression of Low Entropy Strings with Lempel-Ziv Algorithms", *SIAM Journal Comput.*, vol. 29, no. 3, pp. 893-911.

[14] V.N. Potapov, "Redundancy Estimates for the Lempel Ziv Algorithm of Data Compression", *Discrete Applied Mathematics*, vol. 135, pp. 245-254, 2004.

[15] A. Puglisi, E. Caglioti, V. Loreto, A. Vulpiani, "Data Compression and Learning in Time Sequences Analysis", *Physica D: Non Linear Phenomena*, vol. 180, issues 1-2, pp. 92-107, 2003.

[16] J. H. Reif, J. A. Storer, "Optimal Lossless Compression of a Class of Dynamic Sources", *Information Sciences Journal*, no. 135, pp. 87-105, Elseiver, 2001.

[17] S. A. Savari, "Redundancy of the Lempel-Ziv Incremental Parsing Rule", *IEEE Transactions on Information Theory*, vol. 43, no. 1, pp. 9-21, 1977.

[18] E. Shermer, M. Avigal, D. Shapira, "Neural Markovian Predictive Compression: An Algorithm for Online Lossless Data Compression", *DCC 2010*, pp. 209-218.

[19] S. Shanmugasundaram, Robert Lourdusamy, " A Comparative Study Of Text Compression Algorithms", *International Journal of Wisdom Based Computing*, vol. 1 (3), pp. 68-76, 2011.

[20] S. Verdu, Te Sun Han, "The role of the asymptotic equipartition property in noiseless source coding", *IEEE Transactions on Information Theory*, vol. 43, issue 3, 1997.

[21] J. S. Vitter, "Design and Analysis of Dynamic Huffman Codes", *Journal of the Association for Computing Machinery*, vol. 34, no. 4, pp. 825-845, 1987.

[22] A. D. Wyner, J. Ziv, "The Sliding-Window Lempel-Ziv Algorithm is Asymptotically Optimal", *Proceedings of the IEEE*, vol. 82, no. 6, 1994.

[23] A. D. Wyner, J. Ziv, A. J. Wyner, "On the Role of Pattern Matching in Information Theory", *IEEE Transactions on Information Theory*, vol. 44, issue 6, pp. 2045 - 2056, 1998.

[24] L. Yang, R. P. Dick, "On-Line Memory Compression for Embedded Systems", *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 9, issue 3, 2010.

[25] J. Ziv, "The Universal LZ77 Compression Algorithm Is Essentially Optimal for Individual Finite-Length N-Blocks", *IEEE Transactions on Information Theory*, vol. 55, issue 5, 2009.

[26] J. Ziv, A. Lempel, "A Universal Algorithm for Sequential Data Compression", *IEEE Transactions on Information Theory*, vol. IT-24, pp. 337-343, 1977.

[27] J. Ziv, A. Lempel, "Compression of Individual Sequences via Variable-Rate Coding", *IEEE Transactions on Information Theory*, vol. IT-24, pp. 530-536, 1978.