Wireless Netw (2017) 23:2521–2532
DOI 10.1007/s11276-016-1289-9

Improving of entropy adaptive on-line compression

Shlomi Dolev[1] · Sergey Frenkel[2] · Marina Kopeetsky[3] · Muni Venkateswarlu Kumaramangalam[1]

Published online: 26 May 2016

Abstract Since energy efficiency, high bandwidth, and low transmission delay are challenging issues in mobile networks, due to resource constraints, there is a great importance in designing of new communication methods. In particular, lossless data compression may provide high performance under constrained resources. In this paper we present a novel on-line and entropy adaptive compression scheme for streaming unbounded length inputs. The scheme extends the window dictionary Lempel–Ziv compression and is adaptive and 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*), 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 (Ziv in IEEE Trans Inf Theory 55(5):1941–1944, 2009). This adaptivity can be useful for many communication tasks. In particular, providing efficient utilization of energy consuming wireless devices by data compression. Due to the dynamic and non-uniform structure of multimedia data, adaptive approaches for data processing are of special interest. The *EAC* scheme was tested on different types of files (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. We propose techniques intended to control the adaptive on-line compression process by estimating relative entropy between two sequential blocks of data. This approach may enhance performance of the mobile networks.

Keywords On-line compression · Wireless network · Entropy · Compression ratio

Partially supported by Deutsche Telekom, Rita Altura Trust Chair in Computer Sciences, Israeli Internet Association, Israeli Ministry of Science, Lynne and William Frankel Center and Israel Science Foundation (Grant number 428/11). The second author has been partially supported by the Russian Foundation for Basic Research under grant RFBR 15-07-05316. The third author has been partially supported by the internal research program of the Shamoon College of Engineering. An extended abstract of this work was presented at IEEE NCA 2014.

✉ Marina Kopeetsky
marinako@sce.ac.il

[1] Department of Computer Science, Ben-Gurion University of the Negev, 84105 Beer-Sheva, Israel

[2] Institute of Informatics Problems of Federal Research Center "Computer Science and Control" of Russian Academy of Sciences, Moscow, Russia

[3] Department of Software Engineering, Sami-Shamoon College of Engineering, 84100 Beer-Sheva, Israel

1 Introduction

Energy efficiency is one of the most challenging issues in multimedia communication due to wireless device resource constraints, and the high requirements for high bandwidth, and low transmission time. One of the most challenging tasks for communication entities in distributed systems and mobile 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 distributed system becomes more efficient. We seek for adaptive compression schemes, applicable in the mobile environment that provide the best compression (and consequently minimal usage of the battery power) for different and non-uniform data flows.

Common strings are frequently used in mobile communication and dictionary based lossless compression techniques may be very efficient.

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 currently special interest. 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.

1.1 Related work

A universal coding scheme for sequential data compression was first introduced in [1, 2]. Lempel and Ziv introduced a 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 code-words of fixed length over the same alphabet. It has been demonstrated that as the sliding window size (equivalently, the length of a training sequence) 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 [3–5]. As for non-asymptotic coding for finite data streams, some theorems were derived in [6]. 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 [7]. 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 [2] 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.

Different research directions in improving energy efficiency in wireless multimedia networks have been described in [8]. The majority of compression algorithms suitable for resource-constrained systems such as wireless sensor networks and mobile devices are lossy. Analysis and evaluations on energy efficiency in applying these compression algorithms to resource-constrained mobile multimedia transmission systems have been investigated. Nevertheless, the text files were efficiently compressed using the lossless Lempel–Ziv–Welch *LZW* compression algorithm (which is a modification of the dictionary based *LZ78* algorithm). As a result of this experiment 50 % of energy has been saved than compared to transmitting raw text files without compression.

Efficient coding techniques that are motivated by the time-energy trade-off in message transmission between mobile hosts and mobile support stations have been proposed in [9]. The original approach for saving energy by reducing the number of signals sent by the mobile host has been introduced and analyzed. Assuming the synchronization between the mobile host and the mobile support station and that the mobile host sends only bits with value 1 and is silent while a bit value 0 should be sent, the energy consuming modification of the dictionary based *LZ78* compression algorithm has been proposed.

An on-line compression scheme for delay sensitive wireless sensor networks has been introduced in [10]. 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 [10] 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 [11] 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 [12]. *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 [13]. The proposed method is based on the Belief Propagation algorithm in conjunction with the iterative doping algorithm and the inverse Burrows-Wheeler block sorting transformation. It demonstrates that the compression scheme is effective for non-stationary sources. Nevertheless, unlike our *EAC* scheme, the method of [13] totally relies on the source statistics.

However, as it has been shown in [14], ineffective choice and application of data compression scheme can cause a significant energy loss (instead of energy saving). Hence, investigation of new approaches to data compression in mobile environment is a very important issue.

1.2 Our contribution

In this paper we present a novel On-line entropy adaptive compression scheme (*EAC*) for streaming unbounded length inputs, in which, in addition to our previous work [15] we use decision making about the window dictionary size optimization way by using so-called relative entropy [16]. To compress data on-line, we proposed *EAC* scheme based on the sliding window Lempel–Ziv algorithm [1] to individual finite-length of non-overlapping B-blocks of data, where B is the blocks length. We propose techniques intended to control the adaptive on-line compression process by estimation of relative entropy between two sequential blocks of data. *EAC* is an on-line adaptive scheme that tracks the variable entropy ratio of the source and provides optimal compression of fixed-size data blocks on-line without any computational overhead. Specifically, the window dictionary size is changed in an adaptive manner to fit the current best compression ratio for the input. *EAC* 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. *EAC* tracks the sliding window size n_w dynamically 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. *EAC* 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. 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 [17]. This adaptivity can be useful for many communication tasks, in particular, in providing efficient utilization of energy consuming wireless devices by data compression. Due to the dynamic and non-uniform structure of multimedia data, adaptive approach for data processing is of special interest. *EAC* scheme was tested on different types of files (docx, ppt, jpeg, xls) and over synthesized files that were generated as segments of homogeneous Markov Chains. Our experimental results 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. Compared with the recently proposed schemes, the *EAC* scheme has the following advantages.

1.2.1 Optimality of the compression ratio

Currently the best sliding window size for each individual *N*-block, which implies the minimal codeword length, is computed on-line and applied by the *EAC* scheme. Compared with [18], 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 the 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).

1.2.2 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.

1.2.3 Low computational cost combined with fast adaptivity rate

Unlike the dynamic Huffman coding and the *NMPC* schemes [12, 19, 20], 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 [10], 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.

1.2.4 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 [19, 20] require an exponentially large dictionary, in order to approach the optimal compression possible for a given entropy.

1.2.5 Window size optimization process control

The proposed LZ77 per-block based data compression algorithm for networks that perform under power consumption/transmission delay constraints allows us to make a decision about the window dictionary size optimization way by using so-called *relative entropy* [16]. If two adjacent blocks have high relative entropy we may not spend any resources for choice of optimal LZ77 window size. According to [14] the proper compression of short files enhances the performance of mobile networks. This can be useful for different multimedia data with non-uniform structures. In contrast to mentioned above modern approaches to data compression conditioned energy/delay overhead reduction, we consider a situation of rather small compression ratio; when the compression may not reduce significantly the cost of communication energy, whereas the latency can increase as a result of a compression procedure. As a result, the possibility of decision making about compression perspective can be rather useful.

The simulation results, presented in Sect. 4 demonstrate that the *EAC* scheme can perform, in many cases, better and achieves higher compression ratio on-line, compared with the standard *LZ77* compression scheme. Hence, the *EAC* scheme may be plugged in other window based lossless compression schemes (e.g., [10–12, 21]) in order to increase the compression ratio and improve the overall performance of a given compression algorithm.

1.3 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 introduced and analyzed in Sect. 3. Discussion and analysis of the experimental results, in particular the comparison with recent results on data compression and energy consumption in wireless networks appear in Sect. 4. Finally, conclusions can be found in Sect. 5.

2 Preliminaries

Lossless data compression scheme, based on the dictionary method, is the basis for our *EAC* scheme. The *EAC* scheme is based on the sliding window Lempel–Ziv *LZ77* algorithm that was proposed in [1] and further analyzed in [4–6, 22]. In the *LZ77* scheme the dictionary consists of strings from the sliding window presented into recently generated text. Let $(X_{kk=-\infty}^{k=+\infty})$ be a stationary random process with entropy H. $(X_{kk=-\infty}^{k=+\infty})$ 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 [5] 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 [1] 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 [5]. 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: the size of sliding window and the maximal length of phrases [5, 23], entropy of underlying process, length of stationary segments with their entropy estimations in the case of non-stationary process [18].

The Asymptotic Equipartition property Theorem, *AET*, [24], is commonly used. *AET* is a consequence of the law of large numbers and the ergodic theory [25]. AET 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 when being compared 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 [4, 6] 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 [4, 6]. Nevertheless, in these papers the sliding window size is fixed and determined by the constant entropy of the stochastic source.

In our scope, the 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 [5]. 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, a 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 denote 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 of 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 a very small part of the entire file. On the other hand, large B implies a large transmission delay. Yet, the computed optimal n_w is more stable.

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 shifts 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 [6]. 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 into 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 [18] 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 reiterated, that the source entropy is unknown. In the following example, we explain the change in the window size for two interesting cases.

3.1 The EAC scheme with B-bit delay

The detailed description of the scheme is presented in Fig. 1. Let the information source generate a random (possibly 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).

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 $k > 0$. *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 decoding 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 its 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 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 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 opti-
    mal 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)
```

Fig. 1 Entropy adaptive compression scheme

4 Analysis and experimental results

4.1 Experimental comparison of LZ77 versus 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 [15]. 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 and 3 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 [17] 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 [17], as the issue studied is whether it is possible to estimate the entropy exactly using only individual B-blocks. This means that the sliding window size of the *LZ77*-algorithm in [17] is bounded by the value of B.

In fact, [4] 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 [4] and [17] are similar in a sense that both of them deal with the fixed case; as a finite sequence of size N in [17] means that the sliding window, used by the *LZ77* algorithm in [17], cannot be larger than B. Therefore, these theoretical results demonstrate that our practical scheme can provide 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 [17]. 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 $H(X)$). This 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).

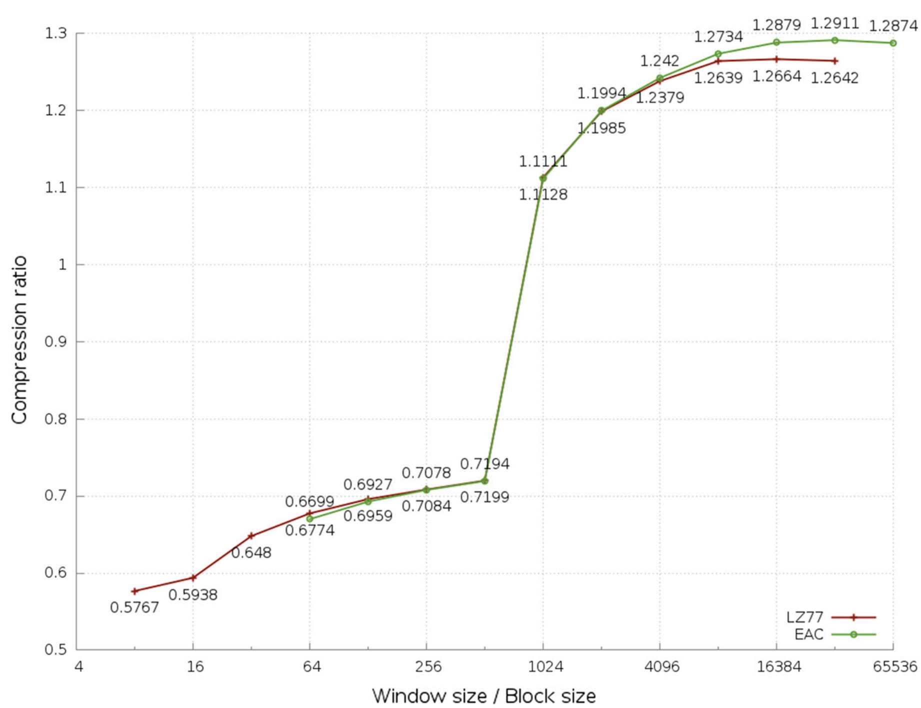

Fig. 2 CR versus window/ block size. File size is 10 Mbit

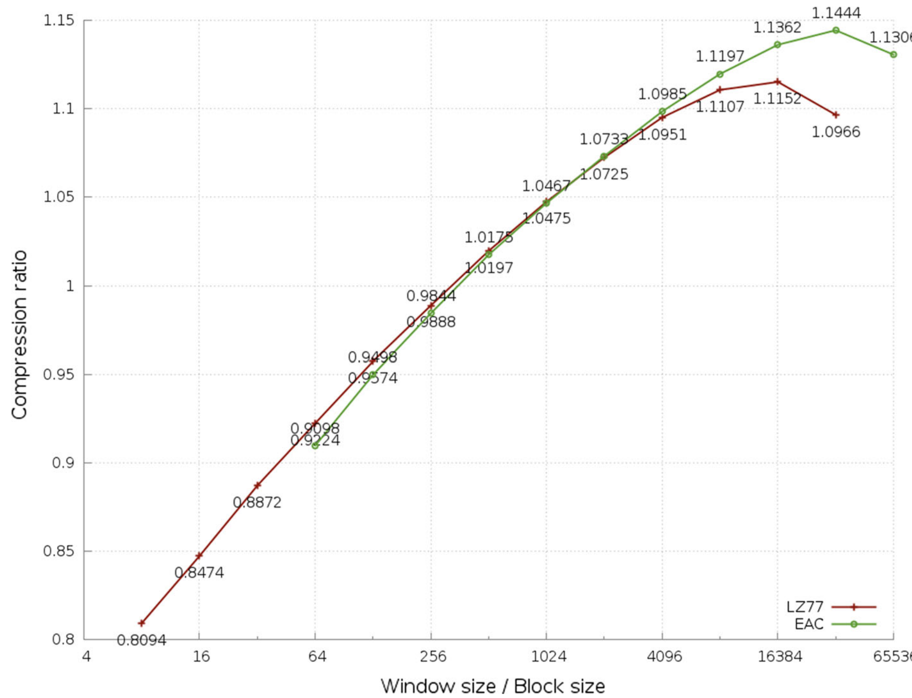

Fig. 3 CR versus window/ block size. File size is 32 K

Generally speaking, an empirical entropy $H_B(N)$, in the theoretical framework of [4], determines the compression ratio of a finite random discrete sequence for non-overlapping B-bit blocks that appear in a sequence of the length $B \cdot M$ bits [17], as this quantity is similar to the classical definition of the empirical entropy of B-blocks in an individual sequence of length $B \cdot M$ [17].

The following parameters that affect 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 [4] 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 [4], 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

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

where B is the size of the portion of bits to be compressed (Sect. 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 [5]. Since our "pyramid based" *EAC* algorithm (Sect. 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, which 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 and 3 demonstrate the impact of the different mentioned above parameters on the compression ratio of the *EAC* scheme (compared with the *LZ77* algorithm).

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 informational properties 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, where 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 [17]. 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 may vary in a 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 [5, 26]. 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 Figs. 2 and 3 demonstrate that the larger file size is, the closer the compression ratio of the *EAC* and the *LZ*77 algorithms.

This is the logical consequence of the expounded above asymptotic property theorem for *LZ* compression for *B*-block sequences [17]. A small improvement ($CR = 1.29$ vs $CR = 1.26$, Fig. 2) of the *EAC* scheme for rather large values of blocks is a result of increasing the lengths of the longest matches, which leads decreasing empirical entropy, and correspondingly, increasing the compression ratio [22]. Besides, the *EAC* scheme can provide higher compression ratio for rather short sequences (Fig. 3), as the asymptotic properties of theorem [17] are still not satisfied for such file sizes by using LZ77. The proper compression for rather small files is very important for enhancing performance of mobile networks [14]. If an original file is large and compression factor is high, compression by any *LZ77* based scheme can save energy. However, if the input file is small, compression factor is smaller due to the training phase.

4.2 Control of the adaptive on-line compression process by estimation of relative entropy

Following [27], the effectiveness of the window based compression may be described by the probability that dictionary substrings are contained in the current portion of B bits.

When the sliding window size n_w in *EAC* (for any B potion) is chosen by the compression criterion, we assume that the portion of the file compressed by using the sliding window, which is a part of the previously compressed portion (Fig. 1) and has almost the same distribution of the longest matches. In general, the optimal window size obtained for the previous portion B_1 of size B, might not be optimal for the next B_2 portion, which uses the previous window as a dictionary [28]. This "non-optimality" means that the average number of bits for each symbol is larger than it can be determined by the Shannon entropy. As a result, the window size optimal for the previous B_1 portion might non be optimal for the next B_2-portion.

Let us consider the following problem: Given a set of training samples from a certain domain, the goal is to compress as accurately as possible new sequences from the same domain. Then the difference between the computed compression ratio and the optimal compression ratio determined by the Shannon Entropy (or equivalently, asymptotic *LZ77* solution) may be characterized as a following phenomenon. Let us assume that the distribution P of a source which emitted the data (Look Ahead *LA*) to be compressed is unknown. Moreover, "training sequence" is distributed according to the certain distribution Q. As a result, the sliding window used to compress *LA* bits is optimal for the data distributed according to the distribution Q. The extra loss in compression quality beyond the entropy (due to the use of Q instead of P) is termed redundancy and is determined by the $n-th$ order Kullback-Leibler (*KL*) divergence (relative entropy) *RE* [28]:

$$RE(n_w||B_2) = E\left(-P^1_{B_2}\log\frac{P^1_{n_w}}{P^1_{B_2}} - (1-P^1_{B_2})\log\frac{1-P^1_{n_w}}{1-P^1_{B_2}}\right)$$

where n_w is the previous sliding window size in the B_1-bit portion (used as a window), B_2 is the next portion (Fig. 1). P^1_B,$P^1_{n_w}$ are the probabilities of "one" in the corresponding portions, E means the averaging. *RE* determines the extra bits per symbol (over the entropy rate) when compressing a sequence distributed according to the distribution Q with a probability measure P.

Table 1 demonstrates the estimation results for the average relative entropy metric *RE* (overall *B*-portions) for four files of the same size $32K$. The empirical entropy of these files is different. Assume that the file source is a Bernoulli process with given probabilities for "zeroes" and "ones". Next, assume that the pyramid base N is equal to the portion size B (Fig. 1). Column *EE* (entropy estimation) determines the range of the entropy among the portions of size B, relative to the *EAC* compression ratio. The *confidence interval* is determined as $confinterval = [confrange_{min},\ confrange_{max}]$, where: $confrange_{min} = |(P^1_B - P^l_B)/P^1_B| \times 100\%$ and $confrange_{max} = |P^1_B - P^r_B)/P^r_B 100| \times 100\%$ are left and right boundaries, respectively, of confidence interval of the probability P^1_B as the Bernoulli trial success probability (of "ones") among all B-portions, relatively to the left (l) and the right (r) bounds of the confidence interval [16]. In order to evaluate the correctness of the compression decision, we must evaluate how the empirical model is close to the theoretical one. For this purpose, we verified a confidence interval of Bernoulli trial

Table 1 Relationship between Relative Entropy *RE* and Compression Ratio *CR*

File	Size	*B*	*CR*	*RE*	*Confrange*(%)	*RE* range	*EE* range
1	32 K	4096	1.65	0.52	21	0.53–0.54	0.83–0.91
		2048	1.972	0.35	29	0.345–0.355	
2		4096	1.73	0.6	20	0.61–0.62	0.77–0.98
		2048	2.2	0.41	22	0.41–0.42	
3		4096	3.7	3.72	3.27	0.61–0.62	0.77–0.98
		2048	4.4	4.95	1.44	0.41–0.42	
4		4096	1.042	0.78	30	0.77–0.79	0.93–0.99
		2048	1.184	0.41	32	0.39–0.41	

probability P_B^1 estimated as a fraction of "ones" in the corresponding portions. In case the confidence interval is rather small, we may consider that our assumption on the Bernoulli distribution for the portion distribution is rather suitable. From Table 1 we can see that the confidence intervals do not exceed 32 % with probability 0.99 (confidence level is $\alpha = 0.01$) [16]. The results of Table 1 demonstrate that the increasing of the relative entropy means the decreased compression ratio for each file. In essence, the deterioration in the compression ratio is caused by the extra bits per symbol wasted due to the different probabilistic measures of two random sequences. The relationship between the relative entropy *RE* and the compression ratio *CR* for different files is more sophisticated, as it depends on the file's entropy (the less entropy the more possible compression ratio).

In essence, the relative entropy can be used as a metric to decide about reason-ability to use the pyramid based optimization of the current B_2-bit portion. Based on the estimation of the relative entropy *RE*, data transmission delay may be significantly reduced. Indeed, the computational time complexity of the *LZ77* for a binary file of size m bits is km for a certain integer k [29]. Many experiments demonstrate [29] that the estimation complexity of the relative entropy for $k > 10$ will be exactly m. We may estimate the relative entropy *RE* regarding the window size n_w, optimal for the previous B_1 portion, and the current Look Ahead B_2 (Fig. 1, lines 14–35). Hence, decision making is in reason-ability to continue the optimization process (Fig. 1, lines 14–35), dependent on the estimated *RE* value. For example, during *EAC* execution with portion size $B = 4096$ (file 4), the relative entropy of each portion (among five portions) is equal to $RE = 0.3506, 0.7197, 0.0247, 1.9768, 1.5468$, and the maximal compression ratio for the corresponding portion, as computed by the *EAC* scheme is $CR = 1.5182, 0.8519,\ 1.3040,\ 0.9390,\ 1.0010$. The more the relative entropy (vector *RE* for each successive portion), the smaller the local compression ratio for this portion. If the relative entropy *RE* of a current portion is high (in comparison with *RE* of other portions), we may not provide proper compression.

For small compression ratio, the compression may not reduce significantly the cost of communication energy, whereas the latency can increase as a result of a compression procedure [30, 31]. As a result, the possibility of decision making about compression perspective can be rather useful.

5 Conclusion

The *LZ77* technique may be effective if the statistics of the dictionary is correct for the remaining encoded sequence (which will asymptotically dominate). As a rule, since the 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, as 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, taking into account this aspect is also a factor for 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 (Fig. 3).

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. Based on [5], the window size is a factor of the *LZ77* overhead. Nevertheless, the larger the fraction of the analyzed compressed sequence (that is used as a dictionary), the greater the probability of finding the longest match in the remaining file. This probability can be increased by coding optimization 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 correct for the rest of encoded sequence (which will asymptotically dominate). This requirement cannot be satisfied for the non-stationary process. In contrast, the *EAC* scheme performs the *LZ77* window-based encoding only for a small B portion of the file.

Acknowledgments We thank Asaf Cohen for useful remarks and discussions. We thank Dmitry Zbarski for implementing and testing our Entropy Adaptive Compression scheme. We thank the Editor and the manuscript reviewers for their invaluable suggestions and comments.

References

1. Ziv, J., & Lempel, A. (1977). A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, *IT–24*, 337–343.
2. Ziv, J., & Lempel, A. (1978). Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, *IT–24*, 530–536.
3. Bender, P. E., & Wolf, J. K. (1991). New asymptotic bounds and improvements on the Lempel–Ziv data compression algorithm. *IEEE Transactions on Information Theory*, *37*(3), 721–729.
4. Hershkovits, Y., & Ziv, J. (1998). On sliding-window universal data compression with limited memory. *IEEE Transactions on Information Theory*, *44*(1), 66–78.
5. Wyner, A. D., & Ziv, J. (1994). The sliding-window Lempel–Ziv algorithm is asymptotically optimal. *Proceedings of the IEEE*, *82*(6), 872–877.
6. Hershkovits, Y., & Ziv, J. (1997). On fixed-database universal data compression with limited memory. *IEEE Transactions on Information Theory*, *43*(6), 1966–1976.
7. Rao Kosaraju, S., Manzini, G. (1999). Compression of low entropy strings with Lempel–Ziv algorithms. *SIAM Journal Computation*, *29*(3), 893–911.
8. Ma, T. (2013). A survey of energy-efficient compression and communication techniques for multimedia in resource constrained systems. *Communications Surveys and Tutorials, IEEE*, *15*(3), 963–972.
9. Dolev, S., Korach, E., & Yukelson, D. (2001). The sound of silence: Guessing games for saving energy in a mobile environment. *Journal of Parallel and Distributed Computing*, *61*, 868–883.
10. Deng, X., & Yang, Y. (2012). On-line adaptive compression in delay sensitive wireless sensor networks. *IEEE Transactions on Computers*, *61*(10), 1429–1442.
11. Kolhe, K. R., Devale, P. R., & Shrivastava, P. (2010). High performance multimedia data compression through improved dictionary. *International Journal of Computer Applications*, *10*(1), 29–35.
12. Shermer, E., Avigal, M., Shapira, D. (2010). Neural Markovian predictive compression: An algorithm for online lossless data compression, *DCC*, pp. 209–218.
13. Caire, G., Shamai, S., Shokrollahi, A., & Verdu, S. (2004). Universal variable-length data compression of binary sources using fountain codes, *IEEE Information Theory Workshop*.
14. Li, Z., Wang, C., Ni, P. (2003). A report on impact of data compression on energy consumption of wireless-networked hand held devices, *Technical Report* no. 03-003, Purdue e-Pubs, Department of Computer Science, Purdue University .
15. Dolev, S., Frenkel, S., Kopeetsky, M., & Zbarski, D. (2014). *Implementation of entropy adaptive online compression, Technical Report 11–03*. Department of Computer Science: Ben Gurion University of the Negev.
16. Rice, J. A. (1995). *Mathematical statistics and data analysis*. North Scituate: Duxbury Press.
17. Ziv, J. (2009). The universal LZ77 compression algorithm is essentially optimal for individual finite-length N-blocks. *IEEE Transactions on Information Theory*, *55*(5), 1941–1944.
18. Reif, J. H., & Storer, J. A. (2001). Optimal lossless compression of a class of dynamic sources. *Information Sciences Journal*, *135*, 87–105. Elseiver.
19. Knuth, D. E. (1985). Dynamic Huffman coding. *Journal of Algorithms*, *6*, 163–180.
20. Vitter, J. S. (1987). Design and analysis of dynamic Huffman codes. *Journal of the Association for Computing Machinery*, *34*(4), 825–845.
21. Yang, L., & Dick, R. P. (2010). On-line memory compression for embedded systems. *ACM Transactions on Embedded Computing Systems (TECS)*, *9*(3), 1–30.
22. Wyner, A. D., Ziv, J., & Wyner, A. J. (1998). On the role of pattern matching in information theory. *IEEE Transactions on Information Theory*, *44*(6), 2045–2056.
23. Potapov, V. N. (2004). Redundancy estimates for the Lempel–Ziv algorithm of data compression. *Discrete Applied Mathematics*, *135*, 245–254.
24. Verdu, S., & Han, T. S. (1997). The role of the asymptotic equipartition property in noiseless source coding. *IEEE Transactions on Information Theory*, *43*(3), 847–857.
25. Hankerson, D., Harris, G. A., & Johnson, P. D. (2003). *Introduction to information theory and data compression* (2nd ed.). Boca Raton: Chapman and Hall/CRC.
26. Bell, T., & Kulp, D. (1993). Longest-match string searching for Lempel–Ziv compression. *Software-Practice and Experience Journal*, *23*(7), 757–771.
27. Baronchelli, A., Caglioti, E., & Loreto, V. (2006). Measuring complexity with zippers, in ArxiVe.
28. Puglisi, A., Caglioti, E., Loreto, V., & Vulpiani, A. (2003). Data compression and learning in time sequences analysis. *Physica D: Non Linear Phenomena*, *180*(1–2), 92–107.
29. Goto, K., Bannai, H. (2013). Simpler and faster Lempel–Ziv factorization, arXiv:1211.3642v2.
30. Savari, S. A. (1977). Redundancy of the Lempel–Ziv incremental parsing rule. *IEEE Transactions on Information Theory*, *43*(1), 9–21.
31. Shanmugasundaram, S., & Lourdusamy, R. (2011). A comparative study of text compression algorithms. *International Journal of Wisdom Based Computing*, *1*(3), 68–76.

Shlomi Dolev Shlomi Dolev received his B.Sc. in Engineering and B.A. in Computer Science in 1984 and 1985, and his M.Sc. and D.Sc. in computer Science in 1990 and 1992 from the Technion Israel Institute of Technology. From 1992 to 1995 he was at Texas A & M University as a visiting research specialist. In 1995 he joined the Department of Mathematics and Computer Science at Ben-Gurion University. Shlomi is the founder and the first Department head of the Computer Science Department at Ben-Gurion University, established in 2000. He is the author of a book entitled Self-Stabilization published by MIT Press in 2000. His publications, more than three hundred conferences, journals and patents include papers in JACM, SIAM journal on computing, Nature Photonics, Nature Communications, Journal of the Optical Society of America A, Distributed Computing, IEEE/ACM Transactions on Networking, ACM

Transactions on Information and System Security, Journal of Cryptology, Journal of Computer and System Sciences, ACM Transactions on Programming Languages and Systems, Neural Computation, and alike. The publications are mostly in the area of distributed computing and communication networks; in particular the self-stabilization property of such systems, cryptography, security, optical computing, brain science and nano-technology. Several agencies and companies support his research including ISF, NSF, IBM (faculty awards), EMC, Intel, Orange France, Deutche Telekom, Israeli Ministries of Science, Economy and Defense and the European Union in the sum of several millions of Dollars. During his stay at Ben-Gurion University Shlomi had visiting positions in several institutions including MIT, Paris 11, Paris 6 and DIMACS. He served in more than a hundred program committees, chairing several including the two leading conferences in distributed computing, DISC 2006, and PODC 2014. Shlomi serves as an Associate Editor in several international journals including the IEEE Transactions on Computers. His research students more than ten PostDocs, over than ten PhD students and twenty MSc students, are positioned in Hi-Tech companies, including IBM, Microsoft, Google and Academia. Currently he is also a co-founderand and a chief scientist officer of the Secret Double Octopus start-up company. Prof. Dolev is the head of the Frankel Center for Computer Science and holds the Ben-Gurion university Rita Altura trust chair in Computer Sciences. From 2011 to 2014, Prof. Dolev served as the Dean of the Natural Sciences Faculty at Ben-Gurion University of the Negev. From 2010 to 2016, he has served as Head of the Inter University Computation Center of Israel. Shlomi currentlty serves as the steering committee head of the Computer Science discipline of the Israel ministry of education.

Sergey Frenkel received his M.S. in Radio communication from Moscow Technical University of Informatics and Communication in 1973, M.S. in Applied Mathematics from Moscow Institute of Elelectronics and Mathematics, and Ph.D. degree in Computer Science from Research Center of Computer Technologies (NICEVT), Moscow, April 1994. He is an associate professor in the Department of Computer Science of Moscow State Technical University "MIREA", and a senior researcher in Institute of Informatics Problems of Federal Research Center "Computer Science and Control" of Russian Academy of Sciences. Now he is also a visiting researcher at Computer Science Department of Ben-Gurion University, Beer-Sheva, Israel.

Marina Kopeetsky received her B.Sc. degree in Mathematics from Sant Persburg Pedagogical University, Russia, in 1989. She repatriated in Israel in 1991 and received M.Sc. and Ph.D. degrees in Mathematics and Computer Science in 1997 and 2003, respectively from Bar Ilan University, Israel. Marina is a senior lecture at the Department of Software Engineering in Shamoon College of Engineering and a researcher at the Department of Computer Science of Ben Gurion University. Her research interests include efficient coding techniques, low overhead cryptography for wireless devices, broadcast encryption and data security. Marina's publications include papers in Ad Hoc Networks, TAAS, Theory of Computation Systems, and alike.

Muni Venkateswarlu Kumaramangalam received his Bachelores degree in Electronics from Sri Venkateswara University, Tirupati, in 2006, and Masters degree in Computer Applications from Anna University, Chennai, in 2009 and Ph.D. degree in Mathematical and Computational Sciences from National Institute of Technology Karnataka, Mangalore, India, in 2016. Since October 2015, he is with Department of Computer Science, Ben-Gurion University of the Negav, Beer Sheva, Israel, where he is a Post Doctoral Fellow working with Prof. Shlomi Dolev. His current research interests include Distributed Computing, Cryptography, Nano Robotics and Wireless Communications.