algorithms

MDPI

Article

Time-Universal Data Compression [†]

Boris Ryabko [1,2]

[1] Institute of Computational Technologies of the Siberian Branch of the Russian Academy of Science, 630090 Novosibirsk, Russia; boris@ryabko.net or b.riabko@g.nsu.ru

[2] Department of Information Technologies, Novosibirsk State University, 630090 Novosibirsk, Russia

[†] The preliminary version of this paper is accepted for ISIT 2019, Paris.

Received: 26 April 2019; Accepted: 27 May 2019; Published: 29 May 2019

Abstract: Nowadays, a variety of data-compressors (or archivers) is available, each of which has its merits, and it is impossible to single out the best ones. Thus, one faces the problem of choosing the best method to compress a given file, and this problem is more important the larger is the file. It seems natural to try all the compressors and then choose the one that gives the shortest compressed file, then transfer (or store) the index number of the best compressor (it requires $\log m$ bits, if m is the number of compressors available) and the compressed file. The only problem is the time, which essentially increases due to the need to compress the file m times (in order to find the best compressor). We suggest a method of data compression whose performance is close to optimal, but for which the extra time needed is relatively small: the ratio of this extra time and the total time of calculation can be limited, in an asymptotic manner, by an arbitrary positive constant. In short, the main idea of the suggested approach is as follows: in order to find the best, try all the data compressors, but, when doing so, use for compression only a small part of the file. Then apply the best data compressors to the whole file. Note that there are many situations where it may be necessary to find the best data compressor out of a given set. In such a case, it is often done by comparing compressors empirically. One of the goals of this work is to turn such a selection process into a part of the data compression method, automating and optimizing it.

Keywords: data compression; universal coding; time-series forecasting

1. Introduction

Nowadays lossless data compressors, or archivers, are widely used in systems of information transmission and storage. Modern data compressors are based on the results of the theory of source coding, as well as on the experience and intuition of their developers. Among the theoretical results, we note, first of all, such deep concepts as entropy, information, and methods of source coding discovered by Shannon [1]. The next important step was done by Fitingoff [2] and Kolmogorov [3], who described the first universal code, as well as Krichevsky who described the first such a code with minimal redundancy [4].

Now practically used data compressors are based on the PPM universal code [5] (which is used along with the arithmetic code [6]), the Lempel–Ziv (LZ) compression methods [7], the Burrows–Wheeler transform [8] (which is used along with the book-stack (or MTF) code [9–11]), the class of grammar-based codes [12,13] and some others [14–16]. All these codes are universal. This means that, asymptotically, the length of the compressed file goes to the smallest possible value (i.e., the Shannon entropy per letter), if the compressed sequence is generated by a stationary source.

In particular, the universality of practically used codes means that we cannot compare their performance theoretically, because all of them have the same limit ratio of compression. On the other hand, the experiments show that the performance of different data compressors depends on a compressed file and it is impossible to single out one of the best or even remove the worst ones. Thus, there is no theoretical or experimental way to select the best data compressors for practical use. Hence, if someone is going to compress a file, he should first select the appropriate data compressor, preferably giving the best compression. The following obvious two-step method can be applied: first, try all available compressors and choose the one that gives the shortest compressed file. Then place a byte representation of its number and the compressed file. When decoding, the decoder first reads the number of the selected data compressor, and then decodes the rest of the file with the selected data compressor. An obvious drawback of this approach is the need to spend a lot of time in order to first compress the file by all the compressors.

In this paper we show that there exists a method that encodes the file with the (close to) optimal compressor, but uses a relatively small extra time. In short, the main idea of the suggested approach is as follows: in order to find the best, try all the compressors, but, when doing it, use for compression only a small part of the file. Then apply the best data compressor for the compression of the whole file. Based on experiments and some theoretical considerations, we can say that under certain conditions this procedure is quite effective. That is why we call such methods "time-universal."

It is important to note that the problems of data compression and time series prediction are very close mathematically (see, for example, [17]). That is why the proposed approach can be directly applied to time series forecasting.

To the best of our knowledge, the suggested approach to data compression is new, but the idea to organize the computation of several algorithms in such a way that any of them worked at certain intervals of time, and their course depends on intermediate results, is widely used in the theory of algorithms, randomness testing and artificial intelligence; see [18–21].

2. The Statement of the Problem and Preliminary Example

Let there be a set of data compressors $F = \{\varphi_1, \varphi_2, ...\}$ and $x_1x_2...$ be a sequence of letters from a finite alphabet A, whose initial part $x_1...x_n$ should be compressed by some $\varphi \in F$. Let v_i be the time spent on encoding one letter by the data compressor φ_i and suppose that all v_i are upper-bounded by a certain constant v_{max}, i.e. $\sup_{i=1,2,...,} v_i \leq v_{max}$. (It is possible that v_i is unknown beforehand.)

The considered task is to find a data compressor from F which compresses $x_1...x_n$ in such a way that the total time spent for all calculations and compressions does not exceed $T(1+\delta)$ for some $\delta > 0$. Note that $T = v_{max}\, n$ is the minimum time that must be reserved for compression and δT is the additional time that can be used to find the good compressor (among $\varphi_1, \varphi_2, ...$). It is important to note that we can estimate δ without knowing the speeds $v_1, v_2, ...$

If the number of data compressors F is finite, say, $\{\varphi_1, \varphi_2, ..., \varphi_m\}$, $m \geq 2$, and one chooses φ_k to compress the file $x_1x_2...x_n$, he can use the following two step procedure: encode the file as $< k > \varphi_k(x_1x_2...x_n)$, where $< k >$ is $\lceil \log m \rceil$-bit binary presentation of k. (The decoder first reads $\lceil \log m \rceil$ bits and finds k, then it finds $x_1x_2...x_n$ decoding $\varphi_k(x_1x_2...x_n)$.) Now our goal is to generalize this approach for the case of infinite $F = \{\varphi_1, \varphi_2, \ldots\}$. For this purpose we take a probability distribution $\omega = \omega_1, \omega_2, ...$ such that all $\omega_i > 0$. The following is an example of such a distribution:

$$\omega_k = \frac{1}{k(k+1)}\,,\ k = 1, 2, 3, ... \tag{1}$$

Clearly, it is a probability distribution, because $\omega_k = 1/k - 1/(k+1)$.

Now we should take into account the length of a codeword which presents the number k, because those lengths must be different for different k. So, we should find such φ_k that the value

$$\lceil -\log \omega_k \rceil + |\varphi_k(x_1 x_2 ... x_n)|$$

is close to minimal. As earlier, the first part $\lceil -\log \omega_k \rceil$ is used for encoding number k (codes achieving this are well-known, e.g., [22].) The decoder first finds k and then $x_1 x_2 ... x_n$ using the decoder corresponding to φ_k. Based on this consideration, we give the following

Definition 1. *We call any method that encodes a sequence $x_1 x_2 ... x_n$, $n \geq 1$, $x_i \in A$, by the binary word of the length $\lceil -\log \omega_j \rceil + |\varphi_j(x_1 x_2 ... x_n)|$ for some $\varphi_j \in F$, a time-adaptive code and denote it by $\hat{\Phi}^{\delta}_{compr}$. The output of $\hat{\Phi}^{\delta}_{compr}$ is the following word:*

$$\hat{\Phi}^{\delta}_{compr}(x_1 x_2 ... x_n) = < \omega_i > \ \varphi_i(x_1 x_2 ... x_n), \tag{2}$$

where $< \omega_i >$ is $\lceil -\log \omega_i \rceil$-bit word that encodes i, whereas the time of encoding is not grater than $T(1+\delta)$ (here $T = v_{max}\, n$).

If for a time-adaptive code $\hat{\Phi}^{\delta}_{compr}$ the following equation is valid

$$\lim_{n \to \infty} \hat{\Phi}^{\delta}_{compr}(x_1 ... x_n)/n = \inf_{1=1,2,...} \lim_{n \to \infty} \varphi_i(x_1 ... x_n)/n \, ,$$

this code is called time-universal.

Comment 1 It will be convenient to reckon that the whole sequence is compressed not letter-by-letter, but by sub-words, each of which, say, a few kilobytes in length. More formally, let, as before, there be a sequence $x_1 x_2 \ldots$, where x_i , $i = 1, 2, ...$ are sub-words whose length (say, L) can be a few kilobytes. In this case $x_i \in \{0,1\}^{8L}$.
Comment 2 Here and below we did not take into account the time required for the calculation of $\log \omega_i$ and some other auxiliary calculations. If in a certain situation this time is not negligible, it is possible to reduce $\hat{T}$ in advance by the required value.

This description and the following discussion are fairly formal, so we give a brief preliminary example of a time-adaptive code. To do this, we took 22 data compressors from [23] and 14 files of different lengths. For each file we applied the following three-step scheme: first we took 1% of the file and sequentially compressed it with all the data compressors. Then we selected the three best compressors, took 5% of the file, and sequentially compressed it with the three compressors selected. Finally, we selected the best of these compressors and compressed the file with this compressor. Thus, the total extra time is limited to 22 × 0.01 + 3 × 0.05 = 0.37, i.e. $\delta \leq 0.37$. Table 1 contains the obtained data.

Table 1. Three-step compression. Extra-time δ = 0.37.

File	Length (bites)	Best Compressor	Chosen Compressor	Chosen/Best (Ratio of Length)
BIB	111,261	nanozip	lpaq8	1.06
BOOK1	768,771	nanozip	nanozip	1
BOOK 2	610,856	nanozip	nanozip	1
GEO	102,400	nanozip	ccm	1.07
NEWS	377,109	nanozip	nanozip	1
OBJ1	21,504	nanozip	tornado	1.23
OBJ2	246,814	nanozip	lpaq8	1.08
PAPER1	53,161	nanozip	tornado	1.52
PAPER2	82,199	nanozip	tornado	1.54
PIC	513,216	zpaq	bbb	1.25
PROGC	39,611	nanozip	tornado	1.42
PROGL	71,646	nanozip	tornado	1.44
PROGP	49,379	lpaq8	tornado	1.4
TRANS	93,695	lpaq8	lpaq8	1

Table 2 shows that the larger the file, the better the compression. The following table gives some insight into the effect of the extra time. Here we used the same three-step scheme, but the size of the parts was 2% and 10% for the first step and the second, respectively, while the extra time was 0.74.

Table 2. Three-step compression. Extra-time δ = 0.74.

File	Legth	Best Compressor	Chosen Compressor	Chosen/Best (Ratio of Length)
BIB	111,261	nanozip	nanozip	1
BOOK1	768,771	nanozip	nanozip	1
BOOK 2	610,856	nanozip	nanozip	1
GEO	102,400	nanozip	nanozip	1
NEWS	377,109	nanozip	lpq1v2	1.14
OBJ1	21,504	nanozip	ccm	1.17
OBJ2	246,814	nanozip	nanozip	1
PAPER1	53,161	nanozip	lpaq8	1.19
PAPER2	82,199	nanozip	nanozip	1
PIC	513,216	zpaq	bbb	1.25
PROGC	39,611	nanozip	lpaq8	1.04
PROGL	71,646	nanozip	lpaq8	1.03
PROGP	49,379	lpaq8	lpaq8	1
TRANS	93,695	lpaq8	lpaq8	1

From the tables it can be seen that the performance of the considered scheme increases significantly when the additional time increases. It worth noting, that if one applied all 22 data compressors to the whole file, the extra time would be 21 instead of 0.74.

3. The Time-Universal Code for the Finite Set of Data Compressors

3.1. Theoretical Consideration

Suppose that there is a file $x_1 x_2 ... x_n$ and data compressors $\varphi_1, ..., \varphi_m$, $n \geq 1, m \geq 1$. Let, as before, v_i be the time spent on encoding one letter by the data compressor φ_i,

$$v_{max} = \max_{i=1,...,n} v_i, \;\; T = n\, v, \tag{3}$$

and let

$$\hat{T} = T(1+\delta)\,,\ \delta > 0. \tag{4}$$

The goal is to find the data compressor φ_j, $j = 1, ..., m$, that compresses the file $x_1 x_2 ... x_n$ in the best way in time $\hat{T}$.

Apparently, the following two-step method is the simplest.

Step 1. Calculate $r = \lfloor \delta T/(m v_{max}) \rfloor$.

Step 2. Compress the file $x_1 x_2 ... x_r$ by φ_1 and find the length of compressed file $|\varphi_1(x_1 ... x_r)|$, then , likewise, find $|\varphi_2(x_1 ... x_r)|$, etc.

Step 3. Calculate $s = \arg\min_{i=1,...,m} |\varphi_i(x_1 ... x_r)|$

Step 4. Compress the whole file $x_1 x_2 ... x_n$ by φ_s and compose the codeword $\langle s \rangle\ \varphi_s(x_1 ... x_n)$, where $\langle s \rangle$ is $\lceil \log m \rceil$-bit word with the presentation of s.

It will be shown that even this simple method is time universal. On the other hand, there are a lot of quite reasonable approaches to build the time-adaptive codes. For example, it could be natural to try a three step procedure, which was considered in the previous part (see Tables 1 and 2), as well as many other versions. Probably, it could be useful to use multidimensional optimization approaches, such as machine learning, so-called deep learning, etc. That is why, we consider only some general conditions needed for time-universality.

Let us give some needed definitions. Suppose, a time-adaptive data-compressor $\hat{\Phi}$ is applied to $x = x_1 ... x_t$. For any φ_i we define

$$\tau_i(t) = \max\{r : \varphi_i(x_1 ... x_r)\ was\ calculated,\ when\ extra\ time\ \delta\, T\ was\ exhausted\}.$$

Theorem 1. *Let there be an infinite word $x_1 x_2 ...$ and time-adaptive method $\hat{\Phi}$ which is based on the finite set of data compressors $\phi_1, ..., \phi_m$. If its additional time of calculation is not grater than δT and the following properties are valid:*

(i) the limits $\lim_{t\to\infty} \varphi_i(x_1 ... x_t)/t$ exist for $i = 1, 2, ..., m$,

(ii) for $i = 1, 2, ..., m$

$$\lim_{t\to\infty} \tau_i(t) = \infty, \tag{5}$$

(iii) for any t the method $\hat{\Phi}(x_1 ... x_t)$ uses such a compressor φ_s for which, for any i

$$(-\log \omega_s + |\varphi_s(x_1 ... x_{\tau_s(t)})|)/\tau_s(t) \le (-\log \omega_i + |\varphi_i(x_1 ... x_{\tau_i(t)})|)/\tau_i(t), \tag{6}$$

Then $\hat{\Phi}(x_1 ... x_n)$ is time universal, that is

$$\lim_{t\to\infty} \hat{\Phi}(x_1 ... x_t)/t\ =\ \inf_{i=1,2,...} \lim_{t\to\infty} |\varphi_i(x_1 ... x_t)|/t \tag{7}$$

A proof is given in the Appendix A, but here we give some informal comments. First, note that property (i) means that any data compressor will participate in the competition to find the best one. Second, if the sequence $x_1 x_2 ...$ is generated by a stationary source and all φ_i are universal codes, then the property (iii) is valid with probability 1 (See, for example, [22]). Hence, this theorem is valid for this case. Besides, note that this this theorem is valid for methods described earlier.

3.2. Experiments

We conducted several experiments to evaluate the effectiveness of the proposed approach in practice. For this purpose we took 20 data compressor from the "squeeze chart (lossless data compression benchmarks)", http://www.squeezechart.com/index.html and files from this site http:

//corpus.canterbury.ac.nz/descriptions/, and http://tolstoy.ru/creativity/90-volume-collection-of-the-works/ (Information about their size is given in the tables below). It is worth noting, that we do not change the collection of the data compressors and the files during experiments. The results are presented in the following tables, where the expression "worst/best" means the ratio of the longest length of the compressed file and the shortest one (for different data compressors). More formally, $worst/best = \max_{i,j=1,\ldots,20}(|\varphi_i|/|\varphi_j|)$. The expression "chosen/best" is a similar value for a chosen data compressor and the best one. The value "chosen/best" is the frequency of occurrence of the event "the best compressor was selected".

Table 3 shows the results of the two-step method, where we took 3% in the first step. Thus, the total extra time is limited to 20 × 0.03 = 0.6, i.e., $\delta \leq 0.6$.

Table 3. Two-step compression. Extra-time δ = 20 × 0.03 = 0.6.

Length of File (byte)	Number of Files	Ratio "Chosen Best"	Average "Worst/best"	Average "Chosen/Best"
$\leq 10^5$	1496	8%	112.87%	103.57%
10^5–10^6	1122	45.72%	131.22%	102.04%
10^6–10^8	384	71%	147.95%	100.99%

Here ratio "chosen best" means a proportion of cases in which the best method was chosen.

Table 4 shows the effect of the extra time δ on the efficiency of the method (In this case we took 5% in the first step).

Table 4. Two-step compression. Extra-time δ = 20 × 0.05 = 1.

Length of File (byte)	Number of Files	Ratio "Chosen Best"	Average "Worst/Best"	Average "Chosen/Best"
$\leq 10^5$	1496	16%	112.87%	102.14%
10^5–10^6	1122	53.63%	131.22%	101.33%
10^6–10^8	384	73%	147.95%	100.84%

Table 5 contains information about the three step method. Here we took 3% in the first step and then took five data compressors with the best performance. Then, in the second step, we tested those five data compressors taking 5% from the file. Hence, the extra time equals $20 \times 0.03 + 5 \times 0.05 = 0.85$.

Table 5. Three-step compression. Extra-time δ = $20 \times 0.03 + 5 \times 0.05 = 0.85$.

Length of File (byte)	Number of Files	Ratio "Chosen Best"	Average "Worst/Best"	Average "Chosen/Best"
$\leq 10^5$	1496	14%	112.87%	102.48%
10^5–10^6	1122	54.9%	131.22%	101.92%
10^6–10^8	384	73%	147.95%	100.86%

Table 6 gives an example of four step method. Here we took 1% in the first step and then took five data compressors with the best performance. Then, in the second step, we tested those five data compressors taking 2% from each file. Basing on the obtained data, we chose three best and tested them on 5% parts. At last, the best of them was used for compression of the whole file. Hence, the extra time equals $20 \times 0.01 + 5 \times 0.02 + 3 \times 0.05 = 0.45$.

Table 6. Four-step compression. Extra-time $20 \times 0.01 + 5 \times 0.02 + 3 \times 0.05 = 0.45$.

Length of File (byte)	Number of Files	Ratio "Chosen Best"	Average "Worst/Best"	Average "Chosen/Best"
$\leq 10^5$	1496	10%	112.87%	103.12%
10^5–10^6	1122	44.69%	131.22%	102.54%
10^6–10^8	384	72%	147.95%	100.88%

If we compare Table 6 and Table 3, we can see that the performance of the four step method is better than two step method, where the extra time is significantly less for the four step method. The same is valid for the considered example of the three step method.

We can see that the three- and four-step methods make sense because they make it possible to reduce the additional time while maintaining the better quality of the method. Also, we can make another important conclusion. All tables show that the method is more efficient for large files. Indeed, the ratio "chosen/best" and the average value "chosen/best" decreases where the file lengths increases. Moreover, the average value "worst/best" increases where the file lengths increases.

4. The Time-Universal Code for Stationary Ergodic Sources

In this section we describe a time-universal code for stationary sources. It is based on optimal universal codes for Markov chains, developed by Krichevsky [4,24] and the twice-universal code [25]. Denote by M_i, $i = 1, 2, ...$ the set of Markov chains with memory (connectivity) i, and let M_0 be the set of Bernoulli sources. For stationary ergodic μ and an integer r we denote by $h_r(\mu)$ the r-order entropy (per letter) and let $h_\infty(\mu)$ be the limit entropy; see for definitions [22].

Krichevsky [4,24] described the codes $\psi_0, \psi_1, ...$ which are asymptotically optimal for $M_0, M_1, ...$, correspondingly. If the sequence $x_1 x_2 ... x_n$, $x_i \in A$, is generated by a source $\mu \in M_i$, the following inequalities are valid almost surely (a.s.):

$$h_i(\mu) \leq |\psi_i(x_1 ... x_t)|/t \ \leq h_i(\mu) + ((|A| - 1)|A|^i + C)/t, \tag{8}$$

where t grows. (Here C is a constant.) The length of a codeword of the twice-universal code ρ is defined as the following "mixture":

$$|\rho(x_1 ... x_t)| = -\log \sum_{i=0}^{\infty} \omega_{i+1} 2^{-|\psi_i(x_1 ... x_t)|}. \tag{9}$$

(It is well-known in information theory [22] that there exists a code with such codeword lengths, because $\sum_{x_1 ... x_t \in A^t} 2^{-|\rho(x_1 ... x_t)|} = 1$.) This code is called twice-universal because for any M_i, $i = 0, 1, ...$, and $\mu \in M_i$ the equality (8) is valid (with different C). Besides, for any stationary ergodic source μ a.s.

$$\lim_{t \to \infty} |\rho_i(x_1 ... x_t)|/t \ = h_\infty(\mu). \tag{10}$$

Let us estimate the time of calculations necessary when using ρ. First, note that it suffices to sum a finite number of terms in (9), because all the terms $2^{-|\psi_i(x_1 ... x_t)|}$ are equal for $i \geq t$. On the other hand, the number of different terms grows, where $t \to \infty$ and, hence, the encoder should calculate $2^{-|\psi_i(x_1 ... x_t)|}$ for growing number i's. It is known [24] that the time spent on coding one letter is close for different codes ψ_i, $i = 0, 1, 2, ...$

Hence, the time spent for encoding one letter by the code ρ grows to infinity, when t grows. The described below time-universal code Ψ^δ has the same asymptotic performance, but the time spent for encoding one letter is a constant.

In order to describe the time-universal code Ψ^δ we give some definitions. Let, as before, v be an upper-bound of the time spent for encoding one letter by any ψ_i, $x_1 ... x_t$ be the generated word,

$$T = t\,v,\; N(t) = \delta T / v = \delta\, t,$$

$$m(t) = \lfloor \log\log N(t) \rfloor,\; s(t) = \lfloor N(t)/(m(t)+1) \rfloor. \tag{11}$$

Denote by Ψ^δ the following method:

Step 1. Calculate $m(t), s(t)$ and

$$|\psi_0(x_1 ... x_{s(t)})|, |\psi_1(x_1 ... x_{s(t)})|, ..., |\psi_{m(t)}(x_1 ... x_{s(t)})|.$$

Step 2. Find such a j that

$$|\psi_j(x_1 ... x_{s(t)})| = \min_{i=0,...,m(t)} |\psi_i(x_1 ... x_{s(t)})|.$$

Step 3. Calculate the codeword $\psi_j(x_1 ... x_t)$ and output

$$\Psi^\delta(x_1 ... x_t) = < j > \psi_j(x_1 ... x_t),$$

where $< j >$ is the $\lceil -\log \omega_{j+1} \rceil$-bit codeword of j. The decoding is obvious.

Theorem 2. *Let $x_1 x_2 ...$ be a sequence generated by a stationary source and the code Ψ^δ be applied. Then this code is time-universal, i.e., a.s.*

$$\lim_{t\to\infty} |\Psi^\delta(x_1 ... x_t)|/t = \inf_{i=0,1,...} \lim_{t\to\infty} |\psi_i(x_1 ... x_t)|/t. \tag{12}$$

Funding: This research was funded by Russian Foundation for Basic Research grant number 18-29-03005.

Conflicts of Interest: The author declares no conflict of interest.

Appendix A

Proof of Theorem 1. Let $\lambda_i = \lim_{t\to\infty} |\varphi_i(x_1 ... x_t)|/t$ and φ_{min} be such a data compressor that $\lambda_{min} = \min_i \lambda_i$. Having taken into account that the set of data compressors F is finite, we can see that for any $\epsilon > 0$ there exists such t_1 that for all $\varphi_i \in F$ and $t > t_1$

$$(\mid \varphi_i(x_1 ... x_t)| - \log \omega_i)/t - \lambda_i| < \epsilon. \tag{A1}$$

From (ii) we obtain that there exists such t_2 that $\tau_i(t_2) > t_1$ for all $i = 1, ..., m$. Let $n \ge t_2$ and $\hat{\Phi}$ be applied to $x_1 x_2 \ldots x_n$. Suppose that a data-compressor φ_s was chosen, when $\hat{\Phi}$ was applied. Hence,

$$(-\log \omega_s + |\varphi_s(x_1 ... x_{\tau_s(n)}|)/\tau_s(n) \le (-\log \omega_{min} + |\varphi_{min}(x_1 ... x_{\tau_{min}(n)}|)/\tau_{min}(n). \tag{A2}$$

From (A1) we can see that

$$(-\log \omega_s + |\varphi_s(x_1 ... x_{\tau_s(n)}|)/\tau_s(n) \ge \lambda_s - \epsilon \tag{A3}$$

and

$$(-\log \omega_{min} + |\varphi_{min}(x_1 ... x_{\tau_{min}(n)}|)/\tau_{min}(n) \le \lambda_{min} + \epsilon. \tag{A4}$$

From the inequalities (A2)–(A4) we obtain $\lambda_s \le \lambda_{min} + 2\epsilon$. Taking into account, that, by definition, $\lambda_{min} \le \lambda_s$ we get

$$\lambda_{min} \le \lambda_s \le \lambda_{min} + 2\epsilon. \tag{A5}$$

Let us estimate $\hat{\Phi}(x_1 ... x_n)/n$. When $\hat{\Phi}(x_1 ... x_n)$ was applied, the data compressor φ_s was chosen. Hence, from (A1) we get

$$\lambda_s - \epsilon \le \hat{\Phi}(x_1 ... x_n)/n \le \lambda_s + \epsilon.$$

From those inequalities and (A5) we can see that

$$\lambda_{min} - \epsilon \le \hat{\Phi}(x_1 ... x_n)/n \le \lambda_{min} + 3\epsilon.$$

It is true for any $\epsilon > 0$, hence, $\lim_{n \to \infty} \hat{\Phi}(x_1 ... x_n)/n = \lambda_{min}$. The theorem is proven. □

Proof of Theorem 2. It is known in Information Theory [22] that $h_r(\mu) \ge h_{r+1}(\mu) \ge h_\infty(\mu)$ for any r and (by definition) $\lim_{r \to \infty} h_r(\mu) = h_\infty(\mu)$. Let $\epsilon > 0$ and r be such an integer that $h_r - h_\infty < \epsilon$. From (11) we can see that there exists such t_1 that $m(t) \ge r$ if $t \ge t_1$. Taking into account (8) and (11), we can see that there exists t_2 for which a.s. $||\psi_r(x_1 ... x_t)|/t - h_r(\mu)| < \epsilon$ if $t > t_2$. From the description of Ψ^δ (the step 3) we can see that there exists such $t_3 > \max\{t_1, t_2\}$ for which a.s.

$$||\psi_r(x_1 ... x_t)|/t - h_\infty(\mu)| \le ||\psi_r(x_1 ... x_t)|/t - h_r(\mu)|$$

$$+(h_r(\mu) - h_\infty(\mu)) < 2\epsilon\,,$$

if $t > t_3$. By definition,

$$|\Psi^\delta(x_1 ... x_t)|/t \le (|\psi_r(x_1 ... x_t)| - \log \omega_{r+1})/t.$$

Having taken into account that ϵ is an arbitrary number and two latest inequalities as well as the fact that a.s. $\inf_{i=0,1,...} \lim_{t \to \infty} |\psi_r(x_1 ... x_t)|/t = h_\infty(\mu)$, we obtain (12). The theorem is proven. □

References

1. Shannon, C. A mathematical theory of communication. *Bell Syst. Tech. J.* **1948**, *27*, 379–423.
2. Fitingof, B.M. Optimal encoding for unknown and changing statistics of messages. *Probl. Inform. Transm.* **1966**, *2*, 3–11.
3. Kolmogorov, A.N. Three approaches to the quantitative definition of information. *Probl. Inform. Transm.* **1965**, *1*, 3–11. [CrossRef]
4. Krichevsky, R. A relation between the plausibility of information about a source and encoding redundancy. *Probl. Inform. Transm.* **1968**, *4*, 48–57.
5. Cleary, J.; Witten, I. Data compression using adaptive coding and partial string matching. *IEEE Trans. Commun.* **1984**, *32*, 396–402. [CrossRef]
6. Rissanen, J.; Langdon, G.G. Arithmetic coding. *IBM J. Res. Dev.* **1979**, *23*, 149–162. [CrossRef]
7. Ziv, J.; Lempel, A. A universal algorithm for sequential data compression. *IEEE Trans. Inf. Theory* **1977**, *23*, 337–343. [CrossRef]
8. A Block-Sorting Lossless Data Compression Algorithm. Available online: https://www.hpl.hp.com/techreports/Compaq-DEC/SRC-RR-124.pdf (accessed on 15 May 2019).
9. Ryabko, B.Y. Data compression by means of a “book stack”. *Probl. Inf. Transm.* **1980**, *16*, 265–269.
10. Bentley, J.; Sleator, D.; Tarjan, R.; Wei, V. A locally adaptive data compression scheme. *Commun. ACM* **1986**, *29*, 320–330. [CrossRef]
11. Ryabko, B.; Horspool, N.R.; Cormack, G.V.; Sekar, S.; Ahuja, S.B. Technical correspondence. *Commun. ACM* **1987**, *30*, 792–797.

12. Kieffer, J.C.; Yang, E.H. Grammar-based codes: A new class of universal lossless source codes. *IEEE Trans. Inf. Theory* **2000**, *46*, 737–754. [CrossRef]
13. Yang, E.H.; Kieffer, J.C. Efficient universal lossless data compression algorithms based on a greedy sequential grammar transform. i. without context models. *IEEE Trans. Inf. Theory* **2000**, *46*, 755–777. [CrossRef]
14. Drmota, M.; Reznik, Y.A.; Szpankowski, W. Tunstall code, Khodak variations, and random walks. *IEEE Trans. Inf. Theory* **2010**, *56*, 2928–2937. [CrossRef]
15. Ryabko, B. A fast on-line adaptive code. *IEEE Trans. Inf. Theory* **1992**, *28*, 1400–1404. [CrossRef]
16. Willems, F.M.J.; Shtarkov, Y.M.; Tjalkens, T.J. The context-tree weighting method: Basic properties. *IEEE Trans. Inf. Theory* **1995**, *41*, 653–664. [CrossRef]
17. Ryabko, B.; Astola, J.; Malyutov, M. *Compression-Based Methods of Statistical Analysis and Prediction of Time Series*; Springer International Publishing: Cham, Switzerland, 2016.
18. Li, M.; Vitanyi, P. *An Introduction to Kolmogorov Complexity and Its Applications*, 3rd ed.; Springer: New York, NY, USA, 2008.
19. Calude, C.S. *Information and Randomness—An Algorithmic Perspective*, 2nd ed.; Springer: Berlin, Germany, 2002.
20. Downey, R.; Hirschfeldt, D.R.; Nies, A.; Terwijn, S.A. Calibrating randomness. *Bull. Symb. Log.* **2006**, *12*, 411–491. [CrossRef]
21. Hutter, M. *Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability*; Springer: Berlin, Germany, 2005.
22. Cover, T.M.; Thomas, J.A. *Elements of Information Theory*; Wiley-Interscience: New York, NY, USA, 2006.
23. Mahoney, M. Data Compression Programs. Available online: http://mattmahoney.net/dc/ (accessed on 15 March 2019).
24. Krichevsky, R. *Universal Compression and Retrival*; Kluwer Academic Publishers: Dordrecht, The Netherlands, 1993.
25. Ryabko, B. Twice-universal coding. *Probl. Inf. Transm.* **1984**, *3*, 173–177.