Re-Pair Achieves High-Order Entropy

Gonzalo Navarro[a], Luís Russo[b]

Deptartment of Computer Science, University of Chile, Chile

Deptartment of Computer Science, University of Lisbon, Portugal

[a]Funded in part by a grant from Yahoo! Research Latin America.
[b]Supported by FCT through the Multiannual Funding Programme for LaSIGE and grant SFRH/BPD/34373/2006.

Abstract

Re-Pair is a dictionary-based compression method invented in 1999 by Larsson and Moffat [LM99, LM00]. Although its practical performance has been established through experiments, the method has resisted all attempts of formal analysis. We show that Re-Pair compresses a sequence $T[1,n]$ over an alphabet of size σ and k-th order entropy H_k, to at most $2nH_k + o(n \log \sigma)$ bits, for any $k = o(\log_\sigma n)$.

Introduction

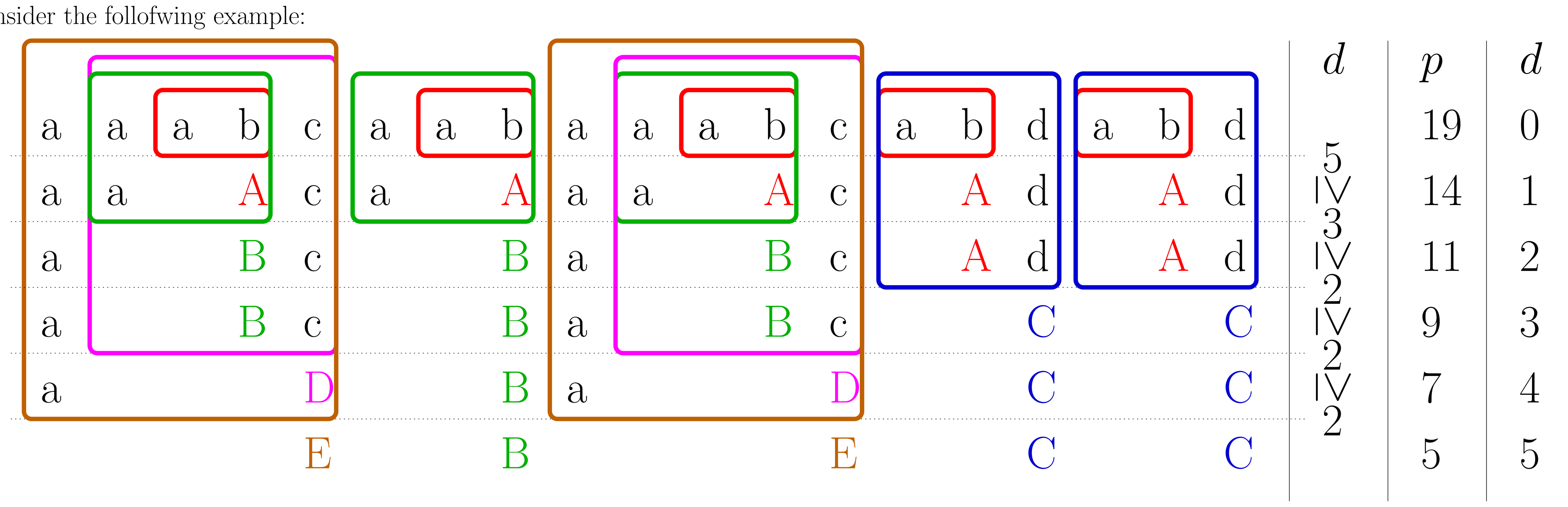
Re-Pair is a dictionary-based compression method invented in 1999 by Larsson and Moffat [LM99, LM00]. As shown by the authors, Re-Pair achieves competitive compression ratios (albeit there are compressors that perform better). However, no theoretical guarantee is given in the original papers, and the method has resisted all attempts of analysis over the years.

We present the first analysis of Re-Pair, proving that it achieves high-order compression. More precisely, given a text $T[1,n]$ over an alphabet of size σ, Re-Pair achieves at most $2nH_k(T) + o(n \log \sigma)$ bits of space, where H_k is the k-th order entropy of T, taken either in the classical information-theory sense over an ergodic source [CT91], or in the sense of empirical entropy [Man01]. Note that, unless $k = 0$, the condition on k implies that the alphabet must be small, $\log \sigma = o(\log n)$.

Analyzing Re-Pair

Re-Pair operates by repeatedly finding the most frequent pair of symbols in T and replaces its occurrences by a new symbol A, adding a rule $A \to ab$ to a dictionary, until every pair appears only once. Re-Pair can be implemented in linear time and space, and it decompresses very fast.

Consider the follofwing example:



The resulting dictionary is:

$$\begin{array}{lll} A & \to a & b \\ B & \to a & A \\ C & \to A & d \\ D & \to B & c \\ E & \to a & D \end{array}$$

We start with a text $T[1,n]$ over an alphabet $\sigma \le n$. Re-Pair compression proceeds in a sequence of steps, each step creating a new dictionary entry. At an arbitrary step of the process, let us call $C = c_1 c_2 \ldots c_p$ the current sequence, mixing original symbols (terminals) and newly created symbols (nonterminals). Hence we call p the length of C (measured in symbols) and d the size of the dictionary (measured in entries, each formed by 2 symbols). In the beginning, $C = T$, $p = n$ and $d = 0$. At each step, d grows by 1 and p decreases at least by 2. Hence d also signals the number of compression steps already executed. Let us call $expand(c_i)$ the sequence of terminals that symbol c_i represents in T ($expand(c_i) = c_i$ if c_i is already terminal). Hence $T = expand(c_1) \cdot expand(c_2) \ldots expand(c_p)$ at any stage. Each $expand(p_i)$ is called a *phrase* and the partition is called a *parsing* of T. We will also denote $expand(XY) = expand(X) \cdot expand(Y)$.

We now point out some properties of Re-Pair. The number of different symbols in the sequence after d steps is at most $\sigma + d$, and thus we would need $\lceil \log(\sigma + d) \rceil$ bits to represent each symbol (by log we mean $\log_2$ in this paper). For simplicity we will make the pessimistic assumption that each dictionary cell, as well as each symbol in the compressed text, will be stored using $\lceil \log n \rceil$ bits. This pessimistic assumption is justified by the next lemma.

Lemma 1. *At any step of the process, it holds $\sigma + d \le n$.*

We speak of "integers" to denote symbols stored using $\lceil \log n \rceil$ bits. The size of the compressed data, at any step of the process, is therefore upper bounded by $(p + 2d)\lceil \log n \rceil$ bits (recall that each dictionary entry occupies 2 integers). The following lemma is rather obvious but important.

Lemma 2. *The size of the compressed data, $p + 2d$ integers, does not increase along the process.*

It is also clear that b cannot increase along the process, as shown in the next lemma.

Lemma 3. *The frequency of the most repeated pair does not increase from one step to the next.*

We will also make use of the following lemma.

Lemma 4. *Let $expand(XY) = expand(ZW)$ for two consecutive pairs of (terminal or nonterminal) symbols in C at any stage of the compression. Then $X = Z$ and $Y = W$.*

Finally, we make heavy use of the following theorem, proved by Kosaraju and Manzini.

Theorem 1. *[KM99] Let $y_1 \ldots y_t$ denote a parsing of the string $T[1,n]$ over an alphabet of size σ, such that each phrase y_i appears at most b times. For any $k \ge 0$ we have*

$$t \log t \quad \le \quad nH_k(T) \; + \; t \log(n/t) \; + \; t \log b \; + \; \Theta(t(1 + k \log \sigma))$$

We are now ready to state our main result.

Theorem 2. *Let $T[1,n]$ be a text over an alphabet of size σ and having k-th order (classical or empirical) entropy $H_k(T)$. Then, compression algorithm Re-Pair achieves a representation using at most $2nH_k(T) + o(n \log \sigma)$ bits for any $k = o(\log_\sigma n)$ (which implies $\log \sigma = o(\log n)$ unless $k = 0$).*

Proof. We study $p + 2d$ when the most frequent pair occurs at most $b = \log^2 n$ times. This is achieved in at most $n/(b+1)$ steps, hence $2d\lceil \log n \rceil < 2(n/b)(\log(n) + 1) = O(n/\log n) = o(n)$. Consider the parsing $expand(c_1 c_2)$, $expand(c_3 c_4)$, … of $t = \lceil p/2 \rceil$ strings that do not appear more than b times and apply lemma 4. Further algebra, especially when $t\lceil \log n \rceil > n/\log n$, gives the bound for $t\lceil \log n \rceil$.

Although it is not the case, imagine we are at the last step in our example. In that case part of the explanation in the previous paragraph can represented schematically as:

$$\begin{array}{ccc} E \;\; B & & \text{aaabc} \;\; \text{aab} \\ \not\Downarrow & & \not\Downarrow \\ E \;\; C & \Rightarrow & \text{aaabc} \;\; \text{abd} \\ \not\Downarrow & \text{(lemma 4)} & \not\Downarrow \\ C & & \text{abd} \end{array}$$

□

The full paper is available as Technical Report TR/DCC-2007-12, Nov 2007, that can obtained from ftp://ftp.dcc.uchile.cl/pub/users/gnavarro/repair-analysis.ps.gz

References

[CT91] T. Cover and J. Thomas. *Elements of Information Theory.* Wiley, 1991.

[KM99] R. Kosaraju and G. Manzini. Compression of low entropy strings with Lempel-Ziv algorithms. *SIAM Journal on Computing*, 29(3):893–911, 1999.

[LM99] J. Larsson and A. Moffat. Off-line dictionary-based compression. In *Proc. DCC*, pages 296–305, 1999.

[LM00] J. Larsson and A. Moffat. Off-line dictionary-based compression. *Proc. IEEE*, 88(11):1722–1732, 2000.

[Man01] G. Manzini. An analysis of the Burrows-Wheeler transform. *Journal of the ACM*, 48(3):407–430, 2001.