N89-22348

PERFORMANCE OF LEMPEL-ZIV COMPRESSORS WITH DEFERRED INNOVATION

Martin Cohn
Computer Science Department
Brandeis University

INTRODUCTION

The noiseless data-compression algorithms introduced by Lempel and Ziv(6,7) parse an input data string into successive substrings each consisting of two parts: The citation, which is the longest prefix that has appeared earlier in the input, and the innovation, which is the symbol immediately following the citation. In "extremal" versions of the LZ algorithm the citation may have begun anywhere in the input; in "incremental" versions it must have begun at a previous parse position. Originally the citation and the innovation were encoded, either individually or jointly, into an output word to be transmitted or stored. Subsequently, it was been speculated by several authors(2,4,5,7) that the cost of this encoding may be excessively high because the innovation contributes roughly lg(A) bits, where A is the size of the input alphabet, regardless of the compressibility of the source. To remedy this excess, they suggested storing the parsed substring as usual, but encoding for output only the citation, leaving the innovation to be encoded as the first symbol of the next substring. Being thus included in the next substring, the innovation can participate in whatever compression that substring enjoys. We call this strategy deferred innovation. It is exemplified in the algorithm described by Welch(5) and implemented in the C program compress that has widely displaced adaptive Huffman coding (compact) as a UNIX system utility.

While compress achieves respectable compression ratios on highly compressible data (say two-to-one or better), it performs poorly, compared to theory and to other versions of LZ compression, on relatively incompressible data. In the extreme of total incompressibility, such as uniform i.i.d. or well encrypted data, compress frequently expands the input by about 45% when the output word size is 12 bits and by about 90% when the output word size is 16, to mention two common options.[2]

These figures stand in contrast to LZ realizations without deferred innovation, where random data are expanded by about 5% for output words of 12 or more bits. The purpose of this paper is to explain the excessive expansion, and implicitly to warn against the use of deferred-innovation compressors on nearly incompressible data.

Suppose a deferred-innovation LZ algorithm operates on a string of b-bit input characters producing B-bit output words,[3] and assume that as in most implementations the dictionary of citations is initialized with all the individual symbols of the input alphabet. For an input string "x y x z . . ." such an algorithm will output B bits for the first "x", and store "xy"; then output B bits for "y", and store "yx"; then output B bits for "x", and store "xz"; and so on. In general, B bits will be output for every position that initiates a novel pair, that is, a pair not seen earlier. What we shall show in the next part of the paper is that if the input length is much less than the square of the alphabet size, the "typical" string of length N has almost N novel pairs, and therefore the output length must be almost NB, and the compression ratio almost B/b. Now, when the input is a string over the alphabet of 256 bytes, the input length would have to be comparable to 2^{16} to avoid this condition; otherwise the compression ratio will likely be close to 12/8 = 1.5 or 16/8 = 2.0 for common choices of B. This is just the behavior mentioned above for the program compress. The "typical" string is generated by a uniform

[2] In default mode, compress refuses to recode a file doomed to expansion.

[3] While ideally this output word would be variable in length, it is easy to show that not much is gained by the complication, so we shall conform to practice and make the wordlength fixed.

independent source over the alphabet, or selected uniformly from among all the possible strings of length N.

We say an ordered pair of consecutive input symbols is novel if and only if the identical ordered pair has not appeared earlier in the input string. Let $S_i(A,N)$ be the number of strings of length N over an alphabet of size A, with a novel pair beginning at position i. The pair beginning in position i can be repeating, like "xx", or nonrepeating, like "xy". Thus $S_i(A,N)$ consists of two parts, according to whether the novel pair at position i is repeating or not; see Figure 1.

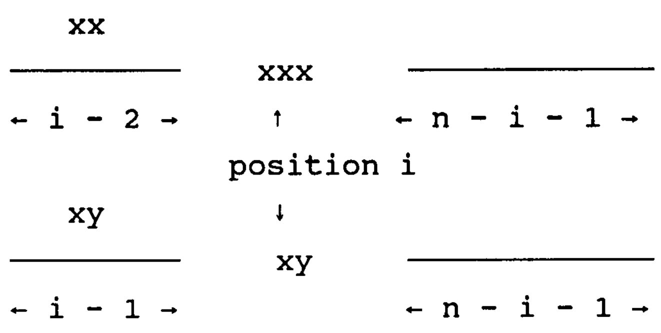

FIGURE 1

Clearly $S_1(A,N) = A^N$, and

$$S_i(A,N) = (A - 1)A^{N-i-1}D(A,i - 2) + (A - 1)A^{N-i}E(A,i - 1),$$

where D(A,i-2) counts strings of length i-2 containing no "xx", and E(A,i-1) counts strings of length i-1 containing no "xy", while x and y range over the alphabet. (We assume the indices are nonnegative and take D(A,0) = E(A,0) = 1 by convention).

Figure 2 shows, for the respective processes (or languages) that contain no repeating pair "xx" or no nonrepeating pair "xy", the state diagrams, adjacency matrices, characteristic equations, and the latters' roots.

$$\begin{bmatrix} A - 1 & 1 \\ A - 1 & 0 \end{bmatrix} \qquad\qquad \begin{bmatrix} A - 1 & 1 \\ A - 2 & 1 \end{bmatrix}$$

$$\delta^2 - (A - 1)\delta - (A - 1) \qquad\qquad \epsilon^2 - A\epsilon + 1$$

$$\delta = \frac{A - 1 \pm \sqrt{(A + 1)^2 - 4}}{2} \qquad\qquad \epsilon = \frac{A \pm \sqrt{A^2 - 4}}{2}$$

FIGURE 2

From linear theory,

$$D(A,k) = d_+\delta_+^k + d_-\delta_-^k \text{ and } E(A,k) = e_+\epsilon_+^k + e_-\epsilon_-^k,$$

where d_+ ,d_- ,e_+ ,and e_- are constants determined by the initial conditions:

$$D_0(A) = E_0(A) = 1,\ D_1(A) = E_1(A) = A.$$

In particular

$$\epsilon_+ = A - A^{-1} - \Theta(A^{-3}),\ \epsilon_- = A^{-1} + \Theta(A^{-3}),$$
$$e_+ = 1 - \Theta(A^{-2}), \text{ and } e_- = \Theta(A^{-2}).$$

We can now estimate $S(A,N)$, the total number of novel pairs among all strings of length N, and $A^{-N}S(A,N)$, the average number of novel pairs per string. We underestimate $S(A,N)$ by ignoring $D(A,N)$, which is positive. Likewise, since e_- and ϵ_-^k are positive, we underestimate by ignoring them. This leaves the approximation

$$S(A,N) \geq A^N + (A-1)A^{N-1}\sum_{i-2}^{N} A^{-i+1}E_{i-1}$$

$$= A^N + (A-1)\sum_{j=1}^{N-1} e_+\epsilon_+^j A^{N-j-1}$$

$$\geq A^N = (A-1)e_+\epsilon_+ [A^{N-2} + A^{N-3}\epsilon + \ldots + \epsilon^{N-2}]$$

Multiplying and dividing by N-1, and using the fact that the arithmetic mean dominates the geometric mean, we have

$$\begin{aligned} S(A,N) &\geq A^N + (A-1)e_+\epsilon_+(N-1)(A\epsilon_+)^{(N-2)/2} \\ &= (1 - \Theta(A^{-1}))\ [A^N + (N-1)A^N\ (1 - 1/A^2)^{(N-2)/2}] \end{aligned}$$

In the limit of large N this last expression is well approximated by

$$A^N + (N-1)A^N \exp(-N + 2/2A^2).$$

Division by A^N confirms the claim that the average number of novel pairs per string of length N remains about N until the string length exceeds the square of the alphabet size.

A simpler but similar calculation can be used to estimate the expected number of novel singletons (symbols) in a string of length N. As before, let $S_i(A,N)$ be the number of sequences in which the ith symbol is novel. Let that symbol be "x"; then the previous i-1 symbols may be anything but x, and the succeeding N-i symbols may be anything at all. Thus

$$S_i(A,N) = (A-1)^{i-1}A^{N-i+1} \text{ and } S(A,N) = \sum_{i=1}^{N} S_i(A,N).$$

As before, this can be underestimated by the geometric mean to give:

$$S(A,N) \geq AN \left(A^{(N-1)/2}(A-1)^{(N-1)/2} \right)$$

which is asymptotically

$$NA^N \exp -(N -1/2A).$$

Thus the average number of novel singletons (symbols) in a sequence of length N remains about N until N exceeds the alphabet size. Similar arguments may be used as well to show that almost all k-tuples will be novel until the sequence length exceeds the kth power of the alphabet size.

DISTRIBUTION OF MEMORY CONTENTS

We next consider the distribution of pairs, triples, and higher-order tuples in the L/Z compressor memory during three regimes: While the memory is filling but not yet full; when it has just filled; and when it is full and in equilibrium. Our assumption is still that input symbols are selected uniformly and independently over some finite alphabet. Another assumption must be made, regarding possible deletions from the memory once it has filled. In practice a variety of deletion strategies have been used, notably l.r.u., whereby the least-recently used entry is deleted to make room for the newest insertion. In this paper we will usually make the simpler assumption that the entry to be deleted is chosen randomly from among the non-singletons. In other words, deletion is random except that the alphabetic symbols are immune.

Memory Filling

Initially the compressor memory (or dictionary) contains α singleton entries, namely the symbols themselves. Each time a match to a singleton is found a pair is inserted; should a pair be matched, its extension to a triple is inserted, and so on. Given the uniform, independent input assumption, it is clear that the likelihood of matching a given pair is only $1/\alpha$ times the likelihood of matching a singleton. Since we are interested mainly in large values of α like 32, 64, 128, 256, we will ignore the possibility of creating quadruples or higher-order tuples, and lump them with the triples. Thus the memory at any time contains α singles, β pairs, and γ others. Let λ be the total number of memory locations, and let $\mu = \lambda$ - alpha be the number of locations available for pairs and higher-order tuples. Then at the time of the t^{th} insertion we have

$$\beta + \gamma = t \text{ for } t < M, \qquad \beta + \gamma = \mu \text{ for } t \geq \mu.$$

The distribution of β (hence γ) at time t during filling is given by the formula

$$\Pr\{\beta | t\} = \alpha^{-t} S(t,\beta)\alpha_{(\beta)}$$

where $S(t,\beta)$ is a Stirling number of the second kind, and $\alpha_{(\beta)}$ is a falling factorial of α^3. The reason is that there are $S_{t,\beta}$ ways of choosing a sequence of t symbols which includes exactly β distinct symbols, and that the identities of those β distinct symbols can be chosen in exactly $\alpha_{(\beta)}$ ways. This count is then divided by α^t, the total number of ways of choosing a sequence of t symbols from the alphabet. Since μ is a large number for any reasonable compressor, we really need the asymptotic distribution in order to analyze the possibly transient behavior when the memory has just filled, but we don't know it at this time.

Transient Period Under L.R.U. Deletion

When the memory has just filled with pairs and higher-order tuples we speculate that there might be interaction between insertion and deletion by the l.r.u. rule that could cause temporary instability.

In particular, if the memory size is close to α^2 then most of the earliest arrivals will have been pairs, and many of the recent arrivals will be triples, as a result of pairs having been matched and extended. This suggests a "gradient" from most recent to least recent shading from triples to pairs. In such a case, under l.r.u.-deletion, disproportionately more pairs will be deleted, abnormally increasing the proportion of non-pairs until the inability to match triples causes pairs to be recreated and reinserted. The alternation in proportions of pairs versus higher-order tuples would likely damp out. This transient behavior has not been confirmed, but is a topic of ongoing research. The distinction between this hypothesis and the equilibrium analyses below stems from the l.r.u. deletion policy, which makes critical not just the distribution of pairs and non-pairs, but their arrangement in memory as well.

Equilibrium State and Distribution

Finally we consider the distribution of memory contents and the compression ratio at equilibrium. We once again invoke the assumption of random deletion (contrary to the l.r.u. rule used in the previous, speculative section). First we solve for an equilibrium state, that is, a ratio of pairs to non-pairs that is stable, and then we generalize to an equilibrium distribution of probabilities of ratios.

Equilibrium State

Suppose that the memory is full, that it contains β pairs (and thus $\mu - \beta$ non-pairs) and that a randomly chosen input pair is read. The probability that the input matches some pair in the memory is β/α^2. The probability that some pair (rather than a triple) is chosen for deletion is β/μ. Since these are independent events, the four joint probabilities for the change in β are:

$$\Delta\beta = \left[\begin{array}{ll} (+1) \cdot \dfrac{\alpha^2 - \beta}{\alpha^2} \cdot \dfrac{\mu - \beta}{\mu} & \text{gain a pair, lose a triple} \\ 0 \cdot \dfrac{\alpha^2 - \beta}{\alpha^2} \cdot \dfrac{\beta}{\mu} & \text{gain a pair, lose a pair} \\ 0 \cdot \dfrac{\beta}{\alpha^2} \cdot \dfrac{\mu - \beta}{\mu} & \text{gain a triple, lose a triple} \\ (-1) \cdot \dfrac{\beta}{\alpha^2} \cdot \dfrac{\beta}{\mu} & \text{gain a triple, lose a pair} \end{array}\right.$$

At equilibrium the first and last probabilities must be equal, and we can solve for β: (Recall that we are ignoring the creation of quadruples or high-orders).

$$\beta = \frac{\alpha^2\mu}{\alpha^2 + \mu}; \qquad \gamma = \mu - \alpha = \frac{\mu^2}{\alpha^2 + \mu}$$

Using these values we can estimate the compression ratio achieved in this equilibrium state by compress, which will output B bits (12 by default) for each 2b bits in a pair can be matched, and will output B bits for only b bits in when a pair cannot be matched. The ratio at equilibrium is thus

$$\rho_{eq} = \frac{(\alpha^2 - \beta)B + \beta B}{(\alpha^2 - \beta)b + 2\beta B} = \frac{\alpha^2 B}{(\alpha^2 + \beta)b} = \frac{(\alpha^2 + \mu)\lg\mu}{(\alpha^2 + 2\mu)\lg\alpha} + O(1/\alpha).$$

From this expression we would expect compress in default mode (with b=8, B=12) to have ρ_{eq} = 1.42, which is quite close to experience.

Equilibrium Distribution

We consider next the equilibrium distribution governing the number of pairs present in the compressor memory. Again we assume that the probabilities of creating quadruples, quintuples, and so on are negligible, so that they can be lumped together with the triples. As usual, the singletons are permanent memory residents.

With memory size μ consider the random variable β describing the number of pairs present. β ranges from 0, when the memory has no pairs, to min (μ,α^2) when either the memory is full of pairs, or all pairs are present. As each parse of the uniform, independent input is made, β may increase by 1 or decrease by 1 (except at the extremes) or stay the same, with the respective probabilities given in Section C1. above[4]. This gives us a Markov process with transition matrix

$$T_{i,j} = \begin{cases} \dfrac{i^2}{\alpha^2\mu} & j = i - 1, \\[2ex] \dfrac{i^2}{\alpha^2} + \dfrac{i}{\mu} + -2\dfrac{i^2}{\alpha^2\mu} & j = i, \\[2ex] \left(1 - \dfrac{i}{\alpha^2}\right)\left(1 - \dfrac{i}{\mu}\right) & j = i + 1, \\[2ex] 0 & \text{otherwise.} \end{cases}$$

Because this is a connected Markov process, it has an equilibrium distribution p^1 which satisfies $pT = p$, or $(pT)_i = p_i$. We show in the Appendix that

[4] This distribution was erroneously described in the Snowbird talk as an Ehrenfest model[1]. As we shall see, it is rather like a componentwise product of two Ehrenfest processes.

$$p_i = \genfrac{}{}{0pt}{}{\alpha^2}{i} \genfrac{}{}{0pt}{}{\mu}{i} \Big/ \sum_{0}^{\min(\alpha^2,\mu)} \genfrac{}{}{0pt}{}{\alpha^2}{i} \genfrac{}{}{0pt}{}{\mu}{i}$$

For a very simple example, let $\alpha^2 = 4$, $\mu = 5$. Then

$$T = \begin{bmatrix} 0 & 20 & 0 & 0 & 0 \\ 1 & 8 & 12 & 0 & 0 \\ 0 & 4 & 10 & 6 & 0 \\ 0 & 0 & 9 & 9 & 2 \\ 0 & 0 & 0 & 16 & 4 \end{bmatrix}$$

$$p(i) = \frac{1}{16}(1,4,6,4,1) \times \frac{1}{31}(1,5,10,10,5) = \frac{1}{126}(1,20,60,40,5).$$

This means that asymptotically the distribution is the product of two Gaussian distributions, with relatively displaced means unless $\alpha^2 = \mu$.

ACKNOWLEDGEMENT

I am grateful for conversations on this work with Ira Gessel and Jim Storer. Dana Goldblatt conjectured the dictionary behavior that led to the study of pair/nonpair distributions.

APPENDIX

It suffices to deal with $\mathbf{q} = \mathbf{p}\, \Sigma\, p(i)$ and to show that

$$(qT)_i = q_{i-1}T_{i-1,i} + q_iT_{i,i} + q_{i+1}T_{i+1,i} = {\alpha^2 \atop i}\quad{\mu \atop i} = q_i .$$

$$(qT)_i = {\alpha^2 \atop i-1}\quad{\mu \atop i-1}\quad 1 - \frac{i-1}{\alpha^2}\quad 1 - \frac{i-1}{\mu}$$

$$+ {\alpha^2 \atop i}\quad{\mu \atop i}\left[\frac{i}{\alpha^2} + \frac{i}{\mu} - \frac{2i^2}{\alpha^2\mu}\right] + {\alpha^2 \atop i+1}\quad{\mu \atop i+1}\quad\frac{(i+1)^2}{\alpha^2\mu}$$

$$= \frac{1}{\alpha^2\mu}\Bigg[\ {\alpha^2 \atop i-1}\quad{\mu \atop i-1}\quad(\alpha^2 - i + 1)(\mu - i + 1)$$

$$+ {\alpha^2 \atop i}\quad{\mu \atop i}(\alpha^2 i + \mu i - 2i^2) + {\alpha^2 \atop i+1}\quad{\mu \atop i+1}\quad(i+1)^2\Bigg]$$

$$= \frac{1}{\alpha^2\mu}\ {\alpha^2 \atop i}\quad{\mu \atop i}\quad[i^2 + \alpha^2 i + \mu i - 2i^2 + (\alpha^2 - i)(\mu - i)]$$

$$= {\alpha^2 \atop i}\quad{\mu \atop i} .$$

Q.E.D.

C-5

REFERENCES

1) Feller, William, "An Introduction to Probability Theory and its Applications", John Wiley & Sons Inc., New York, Vol.1, 1950.

2) Miller, V.S. and M.H. Wegman, "Variations on a Theme by Lempel and Ziv", Combinatorial Algorithms on Words, Springer-Verlag (A. Apostolico and Z. Galil, editors), pp. 131-140, 1985.

3) Riordan, John, "An Introduction to Combinatorial Analysis", John Wiley & Sons, Inc., New York, 1958.

4) Storer, J.A. and T.G. Szymanski, "Data Compression Via Textual Substitution", J.ACM 29, 4, pp. 928-951, 1982.

5) Welch, T.A., "A Technique for High-Performance Data Compression", IEEE Computer 17, 6, pp. 8-19, 1984.

6) Ziv, J. and A. Lempel, "A Universal Algorithm for Sequential Data Compression", IEEE Trans. Inf. Theory IT-23, 3, pp. 337-343, 1977.

7) Ziv, J. and A. Lempel, "Compression of Individual Sequences Via Variable Rate Coding", IEEE Trans. Inf. Theory IT-24, 5, pp. 530-536, 1978.