Data Compression Using a Dictionary of Patterns

Ángel Kuri[1] and José Galaviz[2]

[1] Department of Computing, ITAM.
akuri@itam.mx
[2] Department of Mathematics, Facultad de Ciencias, UNAM.
jgc@fciencias.unam.mx

Abstract. Most modern lossless data compression techniques used today, are based in dictionaries. If some string of data being compressed matches a portion previously seen, then such string is included in the dictionary and its reference is included every time it occurs. A possible generalization of this scheme is to consider not only strings made of consecutive symbols, but more general patterns with gaps between its symbols. In this paper we introduce an off-line method based on this generalization. We address the main problems involved in such approach and provide a good approximation to its solution.
Categories and Subject Descriptors: E.4 [**Coding and Information Theory**]–*data compaction and compression*; I.2.8 [**Artificial Intelligence**]:Problem Solving, Control Methods, and Search–*heuristic methods.*

General Terms: Approximation algorithms, heuristics

Additional Key Words and Phrases: Combinatorial optimization, NP-hardness

1 Introduction

Most of the successful modern lossless compression methods used today are based in dictionaries, such as the Lempel-Ziv family [10]. In these approaches, strings of symbols that occur frequently in the data being compressed (which we call the *sample*) are included in a dictionary. Every occurrence of some of these strings in the sample is further replaced by a reference to its location in the dictionary. If strings in the dictionary are large and frequent enough, this scheme achieves sample compression.

These methods only consider strings of consecutive symbols. In this paper we will introduce a generalization by considering "strings" with gaps, whose symbols are no necessarily consecutive. This generalization will be called *meta-symbol* or pattern in the rest of paper. A similar concept is used in [2] for approximate string matching and is analogous to the concept of *motif* commonly used in computational biology [5].

Advances in Artificial Intelligence Theory
Research on Computing Science 16, 2005, pp. 173-182

In this paper a *meta-symbol* is a sequence of symbols in some alphabet, probably interspersed with *don't-care* symbols or gaps of arbitrary length. The symbol used to specify a gap is not in the alphabet and is called the *gap symbol*. Those symbols in a meta-symbol that are in the alphabet are called *solid symbols* or *defined symbols*. Hence, every string of one or more symbols in the alphabet is a meta-symbol, and every string that begins and ends with symbols in the alphabet, and has an arbitrary sequence of gap symbols and alphabet symbols in between, is also a meta-symbol. We can establish the formal definition of meta-symbol as follows:

Definition 1. Given a finite alphabet of symbols Σ, and a special symbol $\gamma \notin \Sigma$, called *gap symbol*, a meta-symbol p in Σ^+ is any string generated by the regular expression:

$$\Sigma \cup [\Sigma(\Sigma \cup \{\gamma\})^\star \Sigma]$$

There are several features associated with every meta-symbol:

- The *order* of the meta-symbol, denoted $o(p)$, is the number of solid or defined symbols in meta-symbol p.
- The meta-symbol *size*, denoted $|p|$, is the total length of meta-symbol considering both, the solid and gap symbols.

A meta-symbol can be specified in several ways. In this paper our representation is made up by two vectors that describe, respectively, the *contents*, and the *structure* of the meta-symbol. For example, if we use the underscore as the gap symbol, the following is an order 5 and size 10 meta-symbol:

$$p_1 = \texttt{EL\_M\_E\_\_\_A}.$$

that occurs in the string (see figure 1): $S_1 = \texttt{GTELKMIELAMAEPGLARTELVMWEQYYA}$. Such meta-symbol can be specified by:

- The contents vector $\boldsymbol{c}(p)$ of the defined symbols in the meta-symbol p in the order they appear. In our example $\boldsymbol{c}(p_1) = [E, L, M, E, A]$. Obviously, the size (number of entries) of this vector is: $|\boldsymbol{c}(p)| = o(p)$
- The structure vector $\boldsymbol{d}$ of the distances between every consecutive symbol in $\boldsymbol{c}$. In our example $\boldsymbol{d}(p_1) = [1, 2, 2, 4]$. The size of this vector is: $|\boldsymbol{d}(p)| = o(p) - 1$.

We are interested in the search of meta-symbols that occur frequently in longer strings of symbols in Σ. We will call *sample* to any string in Σ^+ where the meta-symbol search takes place. Hence we will require additional features in order to characterize a meta-symbol relatively to the sample. Hence, given a meta-symbol p that occurs in a sample S we define:

- The meta-symbol *occurrence list*, denoted $L(p)$, as the increasing ordered list of absolute positions where p occurs in S (the first symbol in a sample is at position zero). In our previous example $L(p_1) = \{2, 7, 19\}$.

- The *meta-symbol frequency*, denoted $f(p)$, as the number of times meta-symbol p occurs in the sample S. In our example: $f(p_1) = 3$.
- The *rough coverage*, denoted $\mathrm{Cov}(p)$, as the formal number of symbols in S, generated (or covered) by all the occurrences of meta-symbol in the sample, that is the product of frequency times the meta-symbol order. In the example $\mathrm{Cov}(p_1) = 5 \times 3 = 15$.
- The *effective coverage*, denoted $\mathrm{Cov}_{\mathrm{eff}}(p)$, as the actual number of symbols in the sample covered by all the occurrences of meta-symbol. This is the rough coverage defined above, minus the number of symbols in the sample covered more than once by different meta-symbol occurrences. In our example, since the letter `E` in position 7 is covered twice by different occurrences of p_1, we have: $\mathrm{Cov}_{\mathrm{eff}}(p_1) = 14$.

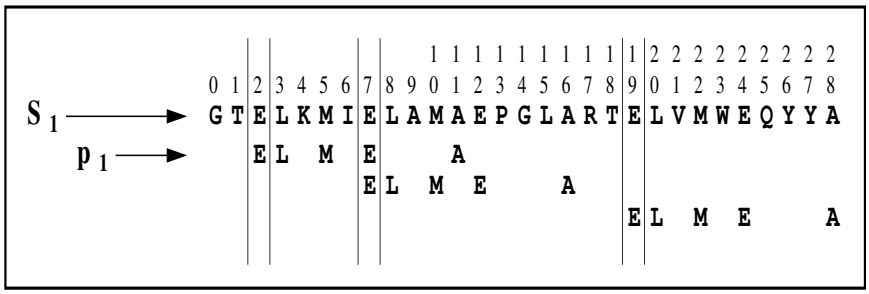

Fig. 1. The meta-symbol p_1 occurs three times in sample S_1.

2 Meta-Symbol-based Data Compression

Our goal is to achieve data compression building a dictionary of frequent meta-symbols found in the sample being compressed. Then every occurrence of a frequent meta-symbol in the sample is replaced by a reference to its position in the meta-symbol dictionary. Since such reference will be shorter than the meta-symbol itself, data compression is achieved.

In order to apply this procedure we need to take several steps:

1. Identify a set P of frequent meta-symbols in a given sample S.
2. Determine the subset $Q \subseteq P$ that maximizes the compression ratio if it is adopted as the dictionary.
3. Determine the best way for meta-symbol representation and encoding, and the best way for reference encoding. The compressed sample will include the, perhaps compressed, dictionary and the compressed representation of the original sample.

In the first step it is needed to find frequent meta-symbols in the sample. The kind of meta-symbols in definition 1 is very general, since the order and gap run-length is not bounded. The ony restriction is imposed by our concept

of *matching*: we use rigid rater than flexible meta-symbols (where the gap run-length can vary in different meta-symbol occurrences). This is a very general, and well known, *pattern discovery* problem [9]. Unfortunately the reported algorithms for the discovery of patterns of the kind we are interested in, have exponential (on the sample size) time complexity [9]. However, if some prerequisites of order and frequency are established, there exist more efficient algorithms. In [6] is introduced the *Teiresias* algorithm, that performs pattern discovery given a minimum frequency and order. The patterns found are called *maximal*. A pattern p is maximal if for every other pattern q, such that p occurs inside q, there exist occurrences of p, where q does not occur. We found this approach very useful and, in order to approximate the first step, we use our own implementation of *Teiresias* algorithm.

The second step is a hard combinatorial optimization problem, in [4] we have shown that it is in the class of NP-complete problems, and here we propose the use of a heuristic in order to approximate the solution. Such approximation will be further optimized by the use of local optimizers (hill climbers).

In the third step, the decisions about the encoding mechanism for the sample re-expression and its dictionary, are guided (as those in the second step) by the the Rissanen minimum description length (MDL) principle [7, 1]. The goal is the simultaneous minimization of data required for the description of the sample being compressed and the model that describes the compression. In our approach we delegate the encoding decision to the compression algorithm, based on the compressed size of the whole set (dictionary + sample re-expression) obtained using different alternatives.

3 Selecting Good Subset of Meta-Symbols

The second step of the aforementioned process is the selection of a good subset of the whole set of frequent meta-symbols found in the sample. The "goodness" of a subset of meta-symbols is given by the compression ratio obtained if such subset of meta-symbols is the dictionary.

Obviously the best subset also must cover a considerably large amount of symbols contained in the sample, since every meta-symbol in the subset must have a good coverage. Also it must have a low amount of data symbols covered several times. That is, the best subset must have efficient meta-symbols: a large amount of covered symbols but a small amount of symbols covered by another meta-symbols or by different occurrences of the meta-symbol itself.

In order to evaluate such characteristic in every proposed meta-symbol we require another concept which we call *exclusive coverage*. Given an order set of meta-symbols Q (a list), the exclusive coverage of some meta-symbol p, denoted $\mathrm{Cov}_{\mathrm{exc}}(p, Q)$, is the total number of symbols in the sample, covered by all the occurrences of p, but not already covered by any other meta-symbol in Q.

Let S be a sample. We will denote by $|S|$ the number of symbols in such sample (its original size). Let P be a set of frequent meta-symbols found in S, and $Q \subseteq P$ a subset of frequent meta-symbols. Let $q' \notin Q$ be the meta-symbol

made of all the symbols in S not already covered by any meta-symbol in Q, therefore the set $D(S,Q) = Q \cup \{q'\}$ is a possible dictionary to express S.

The word bit length needed to refer to any meta-symbol in D is:

$$r = r(Q) = \lceil \log_2 |D(Q)| \rceil = \lceil \log_2 (|Q| + 1) \rceil$$

If $f(q)$ denotes the frequency of meta-symbol $q \in Q$ in the sample S, then the expression of S in terms of dictionary references is:

$$E(Q) = \sum_{q \in D} r\, f(q) = \sum_{q \in Q} r\, f(q) + r = r \left[\sum_{q \in Q} f(q) + 1 \right]$$

Now we need to determine the dictionary size in bits. For every meta-symbol $q \in D(Q)$ we need to specify its $o(q)$ defined symbols and its $o(q) - 1$ gap lengths. Let g be the bit length needed to express the gap runs, and let b be the bit length needed to express every single symbol in D. Therefore the dictionary size in bits is:

$$V(S,Q) = \sum_{q \in D} [b\, o(q) + g\, (o(q) - 1)] + M(S,Q)$$

where M denotes the overhead implicit by the inclusion of a model required to decode the single symbols in the dictionary. If the original symbols are encoded in the dictionary using Huffman encoding, for example, then b will be approximated by the average bit length of Huffman codewords and M is the amount of bits used to store the frequency table required to decode the Huffman codes. In a plain dictionary $M = 0$ and $b = (8\ bits)$.

We will denote with $T(S,Q)$ the size of the compressed sample using the subset of meta-symbols Q, hence:

$$T(S,Q) = V(S,Q) + E(Q)$$

Now we can define the compression ratio:

Definition 2. The *compression ratio* obtained by using the subset of meta-symbols Q is:

$$G(S,Q) = 1 - \frac{T(S,Q)}{|S|} \tag{1}$$

Denoting by p_i the i-th meta-symbol contained in a set of meta-symbols, the coverage of a subset of meta-symbols Q is:

$$\mathrm{Cov}(Q) = \sum_{p_i \in Q} \mathrm{Cov}_{\mathrm{exc}}(p_i, Q \setminus p_i) \tag{2}$$

We can now state our problem as follows:

Definition 3. OptimalMeta-SymbolSubset problem.
Given:

- A data sample S with $|S|$ symbols.
- A set $P = \{p_1, \ldots, p_n\}$ of frequent meta-symbols of S with frequencies $f(p_i) = f_i$ and sizes $t(p_i) = t_i$

Find a subset $Q \subset P$ that maximizes $G(Q)$ subject to the restriction:

$$\mathrm{Cov}(Q) \leq |S| \tag{3}$$

Hence we must find a subset of P. But there are $2^{|P|}$ of such subsets. This is a huge search space even for small values of $|P|$. In fact this is an NP-complete problem as we have shown in [4]. Similar problems have been proven NP-complete in [8, 3]. However in [8] the dictionary size is not considered, and the patterns are strings of consecutive symbols. In [3] also the dictionary size is ignored and coverage of patterns is used as the only criterion to determine the best subset.

Theorem 1. OptimalPatternSubset *Problem is NP-complete.*

Since OPS is NP-complete we need heuristics in order to obtain an approximate solution for large samples. In what follows we will address a proposed heuristic.

4 Heuristics for Subset Selection

4.1 A first proposal.

For the approximate solution of OPS problem we propose a greedy algorithm. Given a set P of frequent meta-symbols found in sample, in each step of our algorithm the meta-symbol with the greatest exclusive coverage is selected and removed from P. This process continues until all the symbols in the sample are covered or the meta-symbols that remain in P do not cover new symbols in the sample. All the meta-symbols selected in each step are stored in a set B, that is the proposed dictionary.

1. Let $B \leftarrow \emptyset$
2. Set the current coverage $Cv \leftarrow \emptyset$.
3. Select the meta-symbol $p \in P$ with highest exclusive coverage $\mathrm{Cov}_{\mathrm{exc}}(p, B)$.
4. Add p to B.
5. Remove p from P
6. Use the coverage of p to update the current coverage Cv.
7. Go to step 3 until $|Cv|$ equals the sample size or $P = \emptyset$ or $\mathrm{Cov}_{\mathrm{exc}}(q, B) = 0$ for every $q \in P$.

With the strategy described we will obtain the subset of meta-symbols B with highest coverage. But a good coverage does not guarantee the best compression ratio. It may occur that meta-symbols with large size and poor frequency or conversely are included in B. Since we want the inclusion of some meta-symbol in the dictionary to be well amortized by its use in the sample, this is not desirable.

4.2 Optimization by local search.

For the improvement of the heuristic described above, we will use hill climbing. Two different hill climbers will be defined:

MSHC *Minimum Step Hill Climber*. Searching the better binary string in the Hamming distance neighborhood of radius 1. Given a binary string that encodes a subset, we perform a search for the best string between those that differ from the given one in only one bit.

MRHC *Minimum Replacement Hill Climber*. Searching the better binary string in the Hamming distance neighborhood of radius 2. Given a binary string, we perform a search for the best string between those that exchanges the position of every bit with value 1 with the position of those bits with value 0.

To find a better subset than the one given by the coverage-based heuristic we sequentially run both hill climbers: the string obtained by the heuristic is passed to MSHC, and the output of this climber is thereafter passed to MRHC. Every hill climber is executed iteratively until no further improvement is obtained or a given number of evaluations is reached.

5 Experimental results

In order to test the effectiveness of the heuristic methods above, and the whole compression algorithm, some experiments were done. We generated a set of random samples with different sizes. Such samples were built using a short meta-symbol alphabet. Every sample was compressed using the method described in this paper, which we call *Pattern-based Data Compression* (PBDC), and other well known methods: Huffman codes (HUF), adaptive Huffman (AHUF), arithmetical encoding (ARIT), Lempel-Ziv-Welch (LZW), and gzip (GZIP), which is a variant of LZ77. Since the samples were built with a pre-defined set of meta-symbols, then we are able to calculate, *a priori*, the compression achieved if such set is defined as the dictionary of meta-symbols.

For every sample size included in the test, we generate as many samples as are needed in order to obtain a confidence interval for the mean of compression ratio with a 95% certainty and a radius below the 5% of mean magnitude. The results obtained are shown in table 1. The leftmost column contains the sample size in bytes, the column "Sam" specifies the number of random samples that were needed, the column labeled "±" contains the radius around the mean value that determines the confidence interval at 95%, "StdDev" stands for the standard deviation. The rightmost column in the table contains the theoretical compression ratio, considering the built-in meta-symbols as the dictionary.

Probably the reader will notice an apparent contradictory result in the first row of the table, since the compression ratio obtained by PBDC is greater than the one predicted by theory in the last column. This phenomenon may occur in short samples, where the accidental combination of dictionary meta-symbols can produce even better meta-symbols with significantly high probability.

Size	Sam	PBDC	±	StdDev	AHUF	HUF	ARIT	LZW	GZIP	Known
128	12	0.340	0.0057	0.0100	0.118	0.070	0.254	0.150	0.001	0.297
256	40	0.506	0.0262	0.0844	0.104	0.080	0.342	0.260	0.202	0.586
384	63	0.601	0.0300	0.1216	0.181	0.163	0.426	0.336	0.311	0.680
512	66	0.680	0.0337	0.1397	0.319	0.314	0.513	0.420	0.430	0.729
1024	67	0.739	0.0357	0.1489	0.228	0.225	0.569	0.434	0.535	0.799
2048	68	0.764	0.0379	0.1596	0.241	0.241	0.679	0.500	0.704	0.836

Table 1. Comparison of PBDC with other compression methods (sample size in bytes).

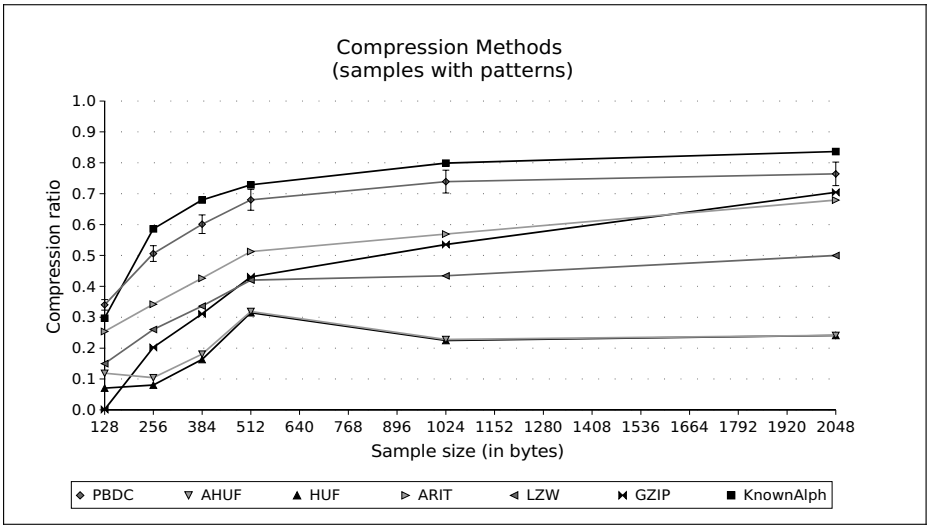

Fig. 2. Comparison of average compression ratios for several algorithms on samples built with meta-symbols.

The data in table 1 are also shown in figure 2, where it is more clear that the PBDC method outperforms all other methods. It is also clear that the compression ratio achieved by PBDC is close to the theoretical one.

6 Summary and Further Work

We have defined a very general compression method based on a dictionary of frequent meta-symbols. The meta-symbols in such dictionary are called *meta-symbols*, since we can think that the sample was generated by a source that produces such meta-symbols instead of the single symbols in the original alphabet. Therefore, working under the hypothesis that the sample is composed by meta-symbols, we try to find its meta-symbolic alphabet or dictionary, made up by those meta-symbols that minimize the expression of the sample together

with the model that describes its composition. Since the search of such meta-alphabet is a hard problem, we propose coverage-based heuristic to approximate its solution.

In order to refine the solution proposed by this heuristic, two different methods of hill climbing where introduced: MSHC and MRHC. These hill climbers are executed on the solution proposed by the heuristic described.

Our results show that the approximation proposed by the heuristic and further refined by the local optimization performed by hill climbers, provides a very good solution for the problem addressed and outperforms several usual data compression algorithms in samples composed by meta-symbols.

Further work is required in two different ways: to test the method with samples not necessarily composed by meta-symbols and to optimize the performance of the meta-symbol discovery algorithm used.

References

[1] Barron, A., J. Rissanen and B. Yu, "The Minimum Description Length Principle in Coding and Modeling", *IEEE Transactions on Information Theory*, 1998, pp. 2743–2760.

[2] Burkhardt, Stefan and Juha Kärkkäinen, "Better Filtering with Gapped q-Grams", *Proceedings of 12th Annual Symposium on Combinatorial Pattern Matching CPM 2001*, Amihood Amir and Gad M. Landau (editors), Lecture Notes in Computer Science, No. 2089, 2001, pp. 73-85.

[3] Klein, Shmuel T., "Improving Static Compression Schemes by Alphabet Extension", *Proceedings of 11th Annual Symposium Combinatorial Pattern Matching CPM 2000*, Raffaele Giancarlo and David Sankoff (editors), Lecture Notes in Computer Science, No. 1848, 2000, pp. 210-221.

[4] Kuri, A. and J. Galaviz, "Pattern-based Data Compression", in *MICAI 2004: Advances in Artificial Intelligence*, Raul Monroy (Editor), Springer Verlag, Lecture Notes in Artificial Intelligence 2972, 2004, pp. 1–10.

[5] Parida, L. *Algorithmic Techniques in Computational Genomics*, Doctoral Dissertation, Courant Institute of Mathematical Sciences, University of New York, 1998.

[6] Rigoutsos, I. and A. Floratos, "Combinatorial Pattern Discovery in biological sequences: The Teiresias Algorithm", *Bioinformatics*, Vol. 14, No. 1, 1998, pp. 55–67.

[7] Rissanen, J., "Modeling by Shortest Data Description", *Automatica*, Vol. 14, 1978, pp. 465–471.

[8] Storer, James y Thomas Szymanski, "Data Compression via Textual Substitution", JACM, Vol. 29, No. 4, october 1982, pp. 928-951.

[9] Vilo, Jaak, *Pattern Discovery from Biosequences*, PhD Thesis, Technical Report A-2002-3, Department of Computer Science, University of Helsinki, 2002.

[10] Ziv, Jacob and Abraham Lempel, "A Universal Algorithm for Sequential Data Compression", *IEEE Transactions on Information Theory*, Vol. 23, No. 3, 1977, pp. 337-343.