Bridging Lossy and Lossless Compression by Motif Pattern Discovery[*]

Alberto Apostolico[†] Matteo Comin[‡] Laxmi Parida [§]

Abstract

We present data compression techniques hinged on the notion of a *motif*, interpreted here as a string of intermittently solid and wild characters that recurs more or less frequently in an input sequence or family of sequences. This notion arises originally in the analysis of sequences, particularly biomolecules, due to its multiple implications in the understanding of biological structure and function, and it has been the subject of various characterizations and study. Correspondingly, motif discovery techniques and tools have been devised. This task is made hard by the circumstance that the number of motifs identifiable in general in a sequence can be exponential in the size of that sequence. A significant gain in the direction of reducing the number of motifs is achieved through the introduction of *irredundant* motifs, which in intuitive terms are motifs of which the structure and list of occurrences cannot be inferred by a combination of other motifs' occurrences. Although suboptimal, the available procedures for the extraction of some such motifs are not prohibitively expensive. Here we show that irredundant motifs can be usefully exploited in lossy compression methods based on textual substitution and suitable for signals as well as text. Actually, once the motifs in our lossy encodings are disambiguated into corresponding lossless codebooks, they still prove capable of yielding savings over popular methods in use. Preliminary experiments with these fungible strategies at the crossroads of lossless and lossy data compression show performances that improve over popular methods (i.e. GZip) by more than 20% in lossy and 10% in lossless implementations.

Keywords Pattern discovery, pattern matching, motif, lossy and lossless data compression, off-line textual substitution, grammar based codes, grammatical inference.

[*]Extended abstracts related to this work were presented at the 2003 and 2004 IEEE Conference on Data Compression, Snowbird, Utah, March 2003 and March 2004, and included in those *Proceedings*. Part of this work was performed while visiting the ZiF Research Center of the University of Bielefeld within the framework of Project "General Theory of Information Transfer and Combinatorics".

[†]Corresponding author. A. Apostolico is with the Department of Computer Sciences, Purdue University, Computer Sciences Building, West Lafayette, IN 47907, USA *and* Dipartimento di Ingegneria dell' Informazione, Università di Padova, Padova, Italy. Work performed in part while on leave at IBM T.J. Watson Center. Supported in part by an IBM Faculty Partnership Award, by NATO Grant PST.CLG.977017, by the Italian Ministry of University and Research under the National Projects "Bioinformatics and Genomics Research" and FIRB RBNE01KNFP, and by the Research Program of the University of Padova. `axa@dei.unipd.it`

[‡]M. Comin is with the Dipartimento di Ingegneria dell' Informazione, Università di Padova, Padova, Italy. `ciompin@dei.unipd.it`

[§]L. Parida is with IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598, USA. `parida@us.ibm.com`

1 Introduction

Data compression methods are partitioned traditionally into lossy and lossless. Typically, lossy compression is applied to images and more in general to signals susceptible to some degeneracy without lethal consequence. On the other hand, lossless compression is used in situations where fidelity is of the essence, which applies to high quality documents and perhaps most notably to textfiles. Lossy methods rest mostly on transform techniques whereby, for instance, cuts are applied in the frequency, rather than in the time domain of a signal. By contrast, lossless textual substitution methods are applied to the input in native form, and exploit its redundancy in terms of more or less repetitive segments or patterns.

When textual substitution is applied to digital documents such as fax, image or audio signal data, one could afford some loss of information in exchange for savings in time or space. In fact, even natural language can easily sustain some degrees of indeterminacy where it is left for the reader to fill in the gaps. The two versions below of the opening passage from the Book1 of the Calgary Corpus, for instance, are equally understandable by an average reader and yet when applied to the entire book the first variant requires 163,837 less bytes than the second one, out of 764,772.

DESCRIPTION OF FARMER OAK – AN INCIDENT When Farmer Oak smile., the corners .f his mouth spread till the. were within an unimportant distance .f his ears, his eye. were reduced to chinks, and ...erging wrinklered round them, extending upon ... countenance li.e the rays in a rudimentary sketch of the rising sun. His Christian name was Gabriel, and on working days he was a young man of sound judgment, easy motions, proper dress, and ...eral good character. On Sundays, he was a man of misty views rather given to postponing, and .ampered by his best clothes and umbrella : upon ... whole, one who felt himself to occupy morally that ... middle space of Laodicean neutrality which ... between the Communion people of the parish and the drunken section, – that ... he went to church, but yawned privately by the t.ime the cong.egation reached the Nicene creed,- and thought of what there would be for dinner when he meant to be listening to the sermon.

DESCRIPTION OF FARMER OAK – AN INCIDENT When Farmer Oak smiled, the corners of his mouth spread till they were within an unimportant distance of his ears, his eyes were reduced to chinks, and diverging wrinkles appeared round them, extending upon his countenance like the rays in a rudimentary sketch of the rising sun. His Christian name was Gabriel, and on working days he was a young man of sound judgment, easy motions, proper dress, and general good character. On Sundays he was a man of misty views, rather given to postponing, and hampered by his best clothes and umbrella : upon the whole, one who felt himself to occupy morally that vast middle space of Laodicean neutrality which lay between the Communion people of the parish and the drunken section, – that is, he went to church, but yawned privately by the time the congregation reached the Nicene creed,- and thought of what there would be for dinner when he meant to be listening to the sermon.

In practice, the development of optimal lossless textual substitution methods is made hard by the circumstance that the majority of the schemes are NP-hard [27]. Obviously, this situation cannot improve with lossy ones. As an approximation, heuristic off-line methods of textual substitution can be based on greedy iterative selection as follows (see e.g., [2, 6, 10]). At each iteration, a substring w of the text x is identified such that encoding a maximal set of non-overlapping instances of w in x yields the highest possible contraction of x; this process is repeated on the contracted textstring, until substrings capable of producing contractions can no longer be found. This may be regarded as inferring a "straight line" grammar [15, 16, 19] by repeatedly finding the production or rule that, upon replacing each occurrence of the "definition" by the corresponding "nonterminal", maximizes the reduction in size of the current textstring representation. Recent implementations of such greedy off-line strategies [6] compare favorably with other current methods, particularly as applied to ensembles of otherwise hardly compressible inputs such as biosequences. They also appear to be the most promising ones in terms of the achievable approximation to optimum descriptor sizes [19].

Off-line methods can be particularly advantageous in applications such as mass production of CD-ROMS, backup archiving, and any other scenario where extra time or parallel implementation may warrant the additional effort imposed by the encoding (see, e.g., [14]).

The idea of trading some amount of errors in reconstruction in exchange for increased compression is ingrained in Rate Distortion Theory [11, 12], and has been recently revived in a number of papers, mostly dealing with the design and analysis of lossy extensions of Lempel-Ziv on-line schemata. We refer to, e.g., [17, 18, 21], and references therein. In this paper, we follow an approach based on the notion of a *motif*, a kind of redundancy emerged particularly in molecular biology and genomic studies. In loose terms, a motif consists of a string of intermittently solid and wild characters, and appearing more or less frequently in an input sequence. Because motifs seem to be implicated in many manipulations of biological as well as more general sequences, techniques for their discovery are of broad interest. We refer to the quoted literature for a more comprehensive discussion. In a nutshell, the role of motifs in our constructions is to capture the auto-correlation in the data by global pattern discovery. The combinatorial structure of our motifs is engineered to minimize redundancy in the "codebook". The presence of a controlled number of don't care characters enhances the compression achievable in the subsequent stage of off-line greedy textual substitution.

In general, the motif discovery and use is made particularly difficult by the fact that the number of candidate motifs in a sequence grows exponentially with the length of that string. Fortunately, a significant reduction in the basis of candidate motifs is possible in some cases. In the context of our textual substitution schemes, for instance, it comes natural to impose that the motif chosen at each iteration satisfies certain maximality conditions that prevent forfeiting information gratuitously. To begin with, once a motif is chosen it seems reasonable to exploit the set of its occurrences to the fullest, compatibly with self-overlaps. Likewise, it seems reasonable to exclude from consideration motifs that could be enriched in terms of solid characters without prejudice in the corresponding set of occurrences.

Recently, a class of motifs called "irredundant" has been identified along these lines that grows linearly with input size [7, 8, 9]. We examine here the application of such motifs to various scenarios of lossy and lossless compression. As it turns out, significant savings can be obtained with this approach.

This paper is organized as follows. In the next section, we recall some basic definitions and properties, and the combinatorial facts subtending to our construction. Section 3 is devoted to the description of our method and the section that follows lists preliminary experiments. Conclusions and plans of future work close the paper.

2 Notions and Properties

Let $s = s_1 s_2 ... s_n$ be a *string* of length $|s| = n$ over an alphabet Σ. We use suf_i to denote the suffix $s_i s_{i+1} ... s_n$ of s and $s[i]$ for the i-th symbol. A character from Σ, say σ, is called a *solid* character and '.' is called a "don't care" character.

Definition 1 *($\sigma_1 \prec, =, \preceq \sigma_2$) If σ_1 is a don't care character then $\sigma_1 \prec \sigma_2$. If both σ_1 and σ_2 are identical characters in Σ, then $\sigma_1 = \sigma_2$. If either $\sigma_1 \prec \sigma_2$ or $\sigma_1 = \sigma_2$ holds, then $\sigma_1 \preceq \sigma_2$.*

Definition 2 *(p occurs at l, Cover) A string, p, on $\Sigma \cup$'.', occurs at position l in s if $p[j] \preceq s[l+j-1]$ holds for $1 \leq j \leq |p|$. String p is said to cover the interval $[l, l+|p|-1]$ on s.*

A *motif* is any element of Σ or any string on $\Sigma \cdot (\Sigma \cup \{.\})^* \cdot \Sigma$.

Definition 3 *(k-Motif m, Location list $\mathcal{L}_m$) Given a string s on alphabet Σ and a positive integer k, $k \leq |s|$, a string m on $\Sigma \cup$ '.' is a motif with location list $\mathcal{L}_m = (l_1, l_2, \ldots, l_q)$, if all of the* *following hold: (1) $m[1], m[|m|] \in \Sigma$, (2) $q \geq k$, and (3) there does not exist a location l, $l \neq l_i$, $1 \leq i \leq q$ such that m occurs at l on s (the location list is of maximal size).*

The first condition ensures that the first and last characters of the motif are solid characters; if don't care characters are allowed at the ends, the motifs can be made arbitrarily long in size without conveying any extra information. The third condition ensures that any two distinct location lists must correspond to distinct motifs.

Using the definition of motifs, the different 2-motifs are as follows: $m_1 = ab$ with $\mathcal{L}_{m_1} = \{1, 5\}$, $m_2 = bc$ with $\mathcal{L}_{m_2} = \{2, 6\}$, $m_3 = cd$ with $\mathcal{L}_{m_3} = \{3, 7\}$, $m_4 = abc$ with $\mathcal{L}_{m_4} = \{1, 5\}$, $m_5 = bcd$ with $\mathcal{L}_{m_5} = \{2, 6\}$ and $m_6 = abcd$ with $\mathcal{L}_{m_6} = \{1, 5\}$.

Notice that $\mathcal{L}_{m_1} = \mathcal{L}_{m_4} = \mathcal{L}_{m_6}$ and $\mathcal{L}_{m_2} = \mathcal{L}_{m_5}$. Using the notation $\mathcal{L} + i = \{x + i | x \in \mathcal{L}\}$, $\mathcal{L}_{m_5} = \mathcal{L}_{m_6} + 1$ and $\mathcal{L}_{m_3} = \mathcal{L}_{m_6} + 2$ hold. We call the motif m_6 *maximal* as $|m_6| > |m_1|, |m_4|$ and $|m_5| > |m_2|$. Motifs m_1, m_2, m_3, m_4 and m_5 are non-maximal motifs.

We give the definition of maximality below. In intuitive terms, a motif m is *maximal* if we cannot make it more specific or longer while retaining the list $\mathcal{L}_m$ of its occurrences in s.

Definition 4 *($m_1 \preceq m_2$) Given two motifs m_1 and m_2 with $|m_1| \leq |m_2|$, $m_1 \preceq m_2$ holds if $m_1[j] \preceq m_2[j + d]$, with $d \geq 0$ and $1 \leq j \leq |m_1|$.*

We also say in this case that m_1 is a *sub-motif* of m_2, and that m_2 *implies* or *extends* or *covers* m_1. If, moreover, the first characters of m_1 and m_2 match then m_1 is also called a *prefix* of m_2. For example, let $m_1 = ab..e$, $m_2 = ak..e$ and $m_3 = abc.e.g$. Then $m_1 \preceq m_3$, and $m_2 \not\preceq m_3$. The following lemma is straightforward to verify.

Lemma 1 *If $m_1 \preceq m_2$ then $\exists\ d \mid \mathcal{L}_{m_1} \supseteq \mathcal{L}_{m_2} + d$, and if $m_1 \preceq m_2$, $m_2 \preceq m_3$, then $m_1 \preceq m_3$.*

Definition 5 *(Maximal Motif) Let m_1, m_2, …, m_k be the motifs in a string s. A motif m_i is maximal in composition if and only if there exists no m_l, $l \neq i$ with $\mathcal{L}_{m_i} = \mathcal{L}_{m_l}$ and $m_i \preceq m_l$. A motif m_i, maximal in composition, is also maximal in length if and only if there exists no motif m_j, $j \neq i$, such that m_i is a sub-motif of m_j and $|\mathcal{L}_{m_i}| = |\mathcal{L}_{m_j}|$. A maximal motif is a motif that is maximal both in composition and in length.*

Requiring maximality in composition and length limits the number of motifs that may be usefully extracted and accounted for in a string. However, the notion of maximality alone does not suffice to bound the number of such motifs. It can be shown that there are strings that have an unusually large number of maximal motifs without conveying extra information about the input.

A maximal motif m is *irredundant* if m and the list $\mathcal{L}_m$ of its occurrences cannot be deduced by the union of a number of lists of other maximal motifs. Conversely, we call a motif m redundant if m (and its location list $\mathcal{L}_m$) can be deduced from the other motifs *without* knowing the input string s. More formally:

Definition 6 *(Redundant motif) A maximal motif m, with location list $\mathcal{L}_m$, is redundant if there exist maximal sub-motifs m_i, $1 \leq i \leq p$, such that $\mathcal{L}_m = \mathcal{L}_{m_1} \cup \mathcal{L}_{m_2} \ldots \cup \mathcal{L}_{m_p}$, (i.e., every occurrence of m on s is already implied by one of the motifs $m_1, m_2, \ldots, m_p$).*

Definition 7 *(Irredundant motif) A maximal motif that is not redundant is called an irredundant motif.*

We use $\mathcal{B}_i$ to denote the set of irredundant motifs in suf_i. Set $\mathcal{B}_i$ is called the *basis* for the motifs of suf_i. Thus, in particular, the basis $\mathcal{B}$ of s coincides with $\mathcal{B}_1$.

Definition 8 *(Basis) Given a sequence s on an alphabet Σ, let $\mathcal{M}$ be the set of all maximal motifs on s. A set of maximal motifs $\mathcal{B}$ is called a basis of $\mathcal{M}$ iff the following hold: (1) for each $m \in \mathcal{B}$, m is irredundant with respect to $\mathcal{B} - \{m\}$, and, (2) let $\mathbf{G}(\mathcal{X})$ be the set of all the redundant maximal motifs generated by the set of motifs $\mathcal{X}$, then $\mathcal{M} = \mathbf{G}(\mathcal{B})$.*

In general, $|\mathcal{M}| = \Omega(2^n)$. The natural attempt now is to obtain as small a basis as possible. Before getting to that, we examine some basic types of maximality.

Lemma 2 *Let m be a maximal motif with no don't care and $|\mathcal{L}_m| = 1$, then $m = s$.*

Proof: Any motif with those properties can be completed into s, by the notion of maximality. ▮

Lemma 3 *Let m be a maximal motif with at least one don't care, then $|\mathcal{L}_m| \geq 2$.*

Proof: Under the hypothesis, it must be $|m| > 1$. The rest is a straightforward consequence of the notion of maximality. ▮

Lemmas 2 and 3 tell us that, other than the string s itself and the characters of the alphabet, the only maximal motifs of interest have more than one occurrence. Solid motifs, i.e., motifs that do not contain any don't care symbol, enjoy a number of nice features that make it pedagogically expedient to consider them separately. Let the equivalence relation $\equiv_s$ be defined on a string s by setting $y \equiv_s w$ if $\mathcal{L}_y = \mathcal{L}_w$. Recall that the *index* of an equivalence relation is the number of equivalence classes in it. The following well known fact from [13] shows that the number of maximal motifs with no don't care is linear in the length of the textstring. It descends from the observation that for any two substrings y and w of s, if $\mathcal{L}_w \cap \mathcal{L}_y$ is not empty then y is a prefix of w or vice versa.

Fact 1 The index k of the equivalence relation $\equiv_x$ obeys $k \leq 2n$.

When it comes to motifs with at least one don't care, it is desirable to obtain as small a basis as possible. Towards this, let x and y be two strings with $m = |x| \leq |y| = n$. The *consensus* of x and y is the string $z_1 z_2 ... z_m$ on $\Sigma \cup$ '.' defined as: $z_i = x_i$ if $x_i = y_i$ and $z_i =$ '.' otherwise $(i = 1, 2, ..., m)$. Deleting all leading and trailing don't care symbols from z yields a (possibly empty) motif that is called the *meet* of x and y. The following general property [7] (cf. proof given in the Appendix) shows that the irredundant 2-motifs are to be found among the pairwise meets of all suffixes of s.

Theorem 1 *Every irredundant 2-motif in s is the meet of two suffixes of s.*

An immediate consequence of Theorem 1 is a linear bound for the cardinality of our set of irredundant 2-motifs: by maximality, these motifs are just some of the $n - 1$ meets of s with its own suffixes. Thus

Theorem 2 *The number of irredundant 2-motifs in a string x of n characters is $O(n)$.*

With its underlying convolutory structure, Theorem 1 suggests a number of immediate ways for the extraction of irredundant motifs from strings and arrays, using available pattern matching with or without FFT. We refer to [1] for a nice discussion of these alternatives. Specific "incremental" algorithms are also available [7] that find all irredundant 2-motifs in time $O(n^3)$. The paradigm explored there is that of iterated updates of the set of base motifs $\mathcal{B}_i$ in a string under consecutive unit symbol extensions of the string itself. Such an algorithm is thus incremental and single-pass, which may lend itself naturally to applications of the kind considered here. The construction used for our experiments must take into account additional parameters related to the density of solid characters, the maximum motif length and minimum allowed number of occurrences. This algorithm is described next.

3 The Pattern Discovery Algorithm

The algorithm follows a standard approach to association discovery: it begins by computing elementary patterns of high *quorum* and then successively extends motifs one solid character at a time until this growth must stop. In general, one drawback of this approach is that the number of patterns at each step grows very rapidly. In our case, the patterns being grown are chosen among $O(n^2)$ substrings of pairwise suffix meets, so that no more than $O(n^3)$ candidates are considered overall. Trimming takes place at each stage, based on the quorum, to keep the overall number of growing patterns bounded. Thus our basis can be detected in polynomial time.

The algorithm makes recurrent use of a routine that solves the following

Set Union Problem, SUP(n, q). Given n sets $S_1, S_2 \ldots, S_n$ on q elements each, find all the sets S_i such that $S_i = S_{i_1} \cup S_{i_2} \cup \ldots \cup S_{i_p}$ $i \neq i_j$, $1 \leq j \leq p$. We present an algorithm in Appendix B to solve this problem in time $O(n^2 q)$.

Input Parameters. The input parameters are: (1) the string s of length n, (2) the quorum k, which is the minimum number of times a pattern must appear, and (3) the maximum number D of consecutive '.' characters allowed in a motif. For convenience in exposition, the notion of a motif is relaxed to include singletons consisting of just one character.

For the rest of the algorithm we will let $m_1.^d m_2$ denote the string obtained by concatenating the elements m_1 followed by d '.' characters followed by the element m_2. Also, recall that $\mathcal{L}_m = \{i | m$ occurs at i on $s\}$.

Computing the basis

The algorithm proceeds in the following steps. M is the set of motifs being constructed.

1. $M = M' \leftarrow \{m' = \sigma \in \Sigma$ and $\mathcal{L}_{m'} \geq k\}$

2. (a) Let $\sigma_i.^d m$, with $0 \leq d \leq D$, denote the left extension of the motif m along a meet. For each motif $m' \in M'$, use meets to compute all of its possible left extensions and store them in the set M''.
   For every $m'' \in M''$, if $|\mathcal{L}_{m''}| < k$ then $M'' \leftarrow M'' - \{m''\}$
   (b) Remove all redundant motifs.
   ```
   For each $m_i \in M$, with $m_i \preceq m''_j$ for some $m''_j \in M''$,
       if $\exists\, m''_{i_1}, m''_{i_2}, \ldots m''_{i_p} \in M''$, $p \geq 1$ such that
           $m_i \preceq m''_{i_j}$ and
           $\mathcal{L}_{m_i} = \mathcal{L}_{m''_{i_1}+f_1} \cup \mathcal{L}_{m''_{i_2}+f_2} \ldots \cup \mathcal{L}_{m''_{i_p}+f_p}$ then
               $M \leftarrow M - \{m_i\}$.
   The above is solved using an instance of the SUP() problem.
   ```
   (c) Update the basis and M'.
   $M \leftarrow M \cup M''$; $M' \leftarrow M''$
3. The previous step is repeated until no changes occur to M.

The algorithm works by iterated extensions of motifs previously in M, where at each iteration a motif is extended (to the left) by zero or more don't cares followed by precisely one solid character. Thus, at the end of the i-th iteration of Step 2, the set M contains motifs with at most $i + 1$ solid characters. As there can be no more that n solid characters in a meet, the number of iterations is bounded by n. Since motifs come from meets at all times, and at most one new motif is considered at one iteration for every ending position on a meet, we have that M' and M'' are each bounded by $O(n^2)$, whereas the elements in M are $O(n^3)$ at all times (in fact, much fewer in practice). At each iteration we have $O(n^2)$ extensions to perform. By solving the $\mathbf{SUP}(n^3, n)$, the algorithm must now try and cover each motif in M by using the new $O(n^2)$ ones in M''. Step 2-b ensures that no motif currently in the set M can be deduced with its location list from the union of other discovered motifs. In other words, the elements of M are irredundant *relative to* M itself, in the sense that no member of M can be inferred from the others. The following claim gives a sharper characterization of the set M.

Theorem 3 *Let $M^{(i)}$ be the set generated by the pattern discovery algorithm at step $i, 0 \leq i \leq n$. Then $M^{(i)}$ contains every k-motif m such that:*

1. *m is a substring of the meet of two suffixes, with at most $i+1$ solid characters and density D.*
2. *m is irredundant relative to the elements of $M^{(i)}$.*

   Moreover,

3. *for every k-motif with these properties not in $M^{(i)}$, there are motifs in $M^{(i)}$ capable of generating it.*
4. *$M^{(i)}$ is a minimum cardinality set with such properties.*

Proof: The claim holds trivially prior to the first iteration. In fact, by initialization $M = M^{(0)} = \{m = \sigma$ is a substring of a meet and m has at least k occurrences $\}$. Clearly, the elements of $M^{(0)}$ are mutually irredundant since they correspond each to a distinct character and hence there is no way to express one of them using the others. $M^{(0)}$ is also exhaustive of such motifs, so that no character of quorum k is left out. At the same time, $M^{(0)}$ is the smallest set capable of generating itself. The first time Step 2 is executed this generates all distinct motifs in the form $\sigma_1 .^d \sigma_2$ that are substrings of meets of quorum k. These motifs are stored in M''. As they are all distinct, they cannot express each other, and the only possibility is for them to obliterate single characters. The latter are now in M', which coincides with M. Through Step 2, a single-character motif $m = \sigma$ is eliminated precisely when it can be synthesized by two-character motifs that either begin or end by σ. As all and only such singletons are eliminated, this propagates all claimed properties. Assuming now the claim true up to step $i - 1 \geq 2$, consider the i-th execution of Step 2. We note that, in an application of Step 2-b, any motif in the set which is used to eliminate motif m must coincide with m on each and every solid character of m. Therefore, no one of the newly introduced motifs with exactly $i+1$ solid characters can be expressed and discarded using *different* motifs with $i+1$ solid characters, or (even worse) motifs with less than i solid characters. Consequently, no such motif can be discarded by this execution of Step 2-b. Also, no collection of motifs formed solely by members of $M^{(h)}$ with $h < i$ can be used to discard other motifs, since, by the operation of the algorithm, any such action would have to have already taken place at some prior iteration. Finally, no mixed collection formed by some new motifs in M'' and motifs currently in M can obliterate motifs in M. In fact, such an action cannot involve only suffixes of the members of M'', or it would have had to be performed at an earlier iteration. Then, it must involve prefixes of motifs in M''. But any such prefix must have been already represented in M, by the third invariant condition.

In conclusion, the only thing that can happen is that motifs currently in M are obliterated by motifs with $i+1$ solid characters, currently in M''. At the beginning of step i, all and only the qualifying k-motifs with i characters are by hypothesis either present directly or generated by motifs in $M^{(i-1)}$, which represents also the smallest possible base for the collection of these motifs. The algorithm extends all the motifs in $M^{(i-1)}$ along a meet of two suffixes, hence all candidate extensions are considered. Now, the net worth of Step 2 is in the balance between the number of newly introduced motifs with $i+1$ solid characters and the number of motifs with i solid characters that get discarded. Assume for a contradiction that a base $\hat{M}$ exists at the outset which is smaller than $M^{(i-1)}$. Clearly, this savings cannot come from a reduction in the number of motifs with $i+1$ solid characters, since eliminating any one of them would leave out a qualifying motif and play havoc with the notion of a base. Hence, such a reduction must come from the elimination of some extra motifs in $M^{(i-1)}$. But we have argued that all and only the motifs in $M^{(i-1)}$ that can be eliminated are in fact eliminated by the algorithm. Thus $M^{(i)}$ must have minimum cardinality. ▮

Theorem 4 *The set M at the outset is unique.*

Proof: Let h be the first iteration such that from $M^{(h-1)}$ we can produce two sets, $M^{(h)}$ and $\bar{M}^{(h)}$, such that $M^{(h)} \neq \bar{M}^{(h)}$ but $|M^{(h)}| = |\bar{M}^{(h)}|$. Clearly, the members of M'' that come from extensions of $M^{(h-1)}$ must belong both to $M^{(h)}$ and $\bar{M}^{(h)}$. Hence the two sets must differ by way of an alternate selection of the members of $M^{(h-1)}$ that are eliminated on behalf of the motifs in M''. But it is clear that any motif that could be, but is not eliminated by M'' will fail to comply with the second clause of the preceding theorem. Hence no option exists in the choice of motifs to be eliminated. ▮

4 Implementation and Experiments

Each phase of our *steepest descent* paradigm alternates the selection of the pattern to be used in compression with the actual substitution and encoding. The sequence representation at the outset is finally pipelined into some of the popular encoders and the best one among the overall scores thus achieved is retained. By its nature, such a process makes it impossible to base the selection of the best motif at each stage on the actual compression that will be conveyed by this motif in the end. The decision performed in choosing the pattern must be based on an estimate, that also incorporates the peculiarities of the scheme or rewriting rule used for encoding. In practice, we estimate at $\log i$ the number of bits needed to encode the integer i (we refer to, e.g., [5] for reasons that legitimate this choice). In one scheme (hereafter, $Code_1$) [6], we eliminate all occurrences of m, and record in succession m, its length, and the total number of its occurrences followed by the actual list of such occurrences. Letting $|m|$ denote the length of m, f_m the number of occurrences of m in the textstring, $|\Sigma|$ the cardinality of the alphabet and n the size of the input string, the compression brought about by m is estimated by subtracting from the $f_m|m|\log|\Sigma|$ bits originally encumbered by this motif on s, the expression $|m|\log|\Sigma| + \log|m| + f_m \log n + \log f_m$ charged by encoding, thereby obtaining:

$$G(m) = \qquad (1)$$
$$(f_m - 1)|m|\log|\Sigma| - \log|m| - f_m \log n - \log f_m.$$

This is accompanied by a fidelity loss $L(m)$ represented by the total number of don't cares introduced by the motif, expressed as a fraction of the original length. If d such gaps were introduced, this would be:

$$L(m) \quad = \quad \frac{f_m d \log|\Sigma|}{f_m|m|\log|\Sigma|} \quad = \quad \frac{d}{|m|}. \qquad (2)$$

Other encodings are possible (see, e.g., [6]). In one scheme (hereafter, $Code_2$), for example, every occurrence of the chosen pattern m is substituted by a pointer to a common dictionary copy, and we need to add one bit to distinguish original characters from pointers. The space originally occupied by m on the text is in this case $(\log|\Sigma| + 1)f_m|m|$, from which we subtract $|m|\log|\Sigma| + \log|m| + \log|f_m| + f_m(\log D + 1)$, where D is the size of the dictionary, in itself a parameter to be either fixed a priori or estimated.

The tables and figures below were obtained from preliminary experiments. The major burden in computations is posed by the iterated updates of the motif occurrence lists, that must follow the selection of the best candidate at each stage. This requires maintaining motifs with their occurrences in a doubly linked list as in Fig. 1: following each motif selection, the positions of the text covered by its occurrences are scanned horizontally. Next, proceeding vertically from each such position, the occurrences of other motifs are removed from their respective lists.

To keep time manageable, most of the experiments were based on a small number of iterations, typically in the range 250-3,000. For Book1, for instance, more than $30k$ motifs could be extracted. Each one of these would convey some compression if used, but time constraints allowed only less than 10% to be implemented. In the pattern discovery stage, a maximum length for motifs was enforced at about 40 to 50 characters, and a threshold of 5 or 6 was put on the overall number of don't care allowed in a single motif, irrespective of its length. The collection of these measures made it possible to test the method on a broad variety of inputs. By the same token, the resulting scores represent quite a conservative estimate of its potential.

The tables summarize scores related to various inputs under various acceptances of loss. Table 1 refers to 8-bit grey-level images as a function of the don't care density allowed (last column). The next table, Table 2 shows results on black and white pictures. These are similar except in this case the loss of one bit translates into that of 1 byte. By their nature, binary or dithered images such as in faxsimile transmission seem to be among the most promising applications of our method. At the same time, it has already been reported that "directional" lossy textual substitution methods can compete successfully even with chrominance oriented methods like JPEG [1]. In view of the results in [6], off-line lossy variants of the kind presented here should perform just as well and probably better. Table 3 shows results for musical records sampled at 8 bits. For this family of inputs, the motif extraction phase alone seems to present independent interest in applications of contents-based retrieval.

Tables 5, 6, and 7 cover inputs from the Calgary Corpus and some yeast families. DNA sequences represent interesting inputs for compression, in part because of the duality between compression and structural inference or classification, in part due to the well know resiliency of bio-sequences towards compression (see, e.g., [6] and references therein). Particularly for text (we stress that lossy compression of bio-sequences is a viable classification tool), lossy compression may be not very meaningful without some kind of reconstruction. As suggested at the beginning of the paper, this might be left to the user in some cases. Otherwise, Table 4 list results obtained by exact completions of the motifs involved in implementation of all of our lossy schemata. It suggests that the bi-lateral context offered by motifs lends itself to better *predictors* than the traditional ones based on the left context alone. In any case, the iteration of motif extraction at several consecutive levels of hierarchy unveils structural properties and descriptors akin to unconventional grammars.

We use Figure 2 to display encodings corresponding to the images from Table 1. The single most relevant parameter here is represented by the density of don't care, which is reported in the last column of the table and also evidenced by the black dots injected in figures at the last column. As mentioned, the maximum length of motifs extracted had to be limited by practical considerations. Even so, it was found that images rarely produce motifs longer than a few tens of characters. More severe consequences of these practical restrictions came from the need to limit the number of motifs actually deployed in compression, which was kept at those with at least 5 to 10 occurrences, corresponding to a quite limited dictionary of 1,000 to 2,000 entries. Interpolation was carried out by averaging from the two solid characters adjacent to each gap. The corresponding discrepancies from the original pixel values reach into 16% in terms of % *number* of inexact pixels, but was found to be only a few percentage points if the variation in *value* of those pixels was measured instead as a percentage of the affected pixels (next to last column of Table 8, and entirely negligible (a fraction of a percent, see last column in Table 8) when averaged over all pixels. This is demonstrated in the reconstructed figures, that show little perceptible change.

As mentioned, our main interest was testing the breadth of applicability of the method rather that bringing it to the limit on any particular class of inputs. This is the scope of future work. In the experiments reported here, the number of iterations (hence, motifs selected or vocabulary size) was in the range of 250 to 1,000 and slightly higher (3,000) for the Calgary Corpus. The length of motifs was limited to few tens of characters and their minimum number of occurrences to 20 or higher. Typically, motifs in the tens of thousands were excluded from consideration on these grounds, which would have been provably capable of contributing savings.

Table 1: Lossy compression of gray-scale images (1 pixel = 1 byte).

file	file len	GZip len [%compr]	$Codec_2$ [%compr]	$Codec_1$ [%compr]	%Diff gzip	%loss	'.'/ char
bridge	66336	$61657_{[7.05]}$	$60987_{[8.06]}$	$57655_{[13.08]}$	6.49	0.42	1/4
			$60987_{[8,06]}$	$50656_{[23.63]}$	17.84	14.29	1/3
camera	66336	$48750_{[26.51]}$	$47842_{[27.88]}$	$46192_{[30.36]}$	5.25	0.74	1/6
			$48044_{[27.57]}$	$45882_{[30.83]}$	5.88	2.17	1/5
			$47316_{[28.67]}$	$43096_{[35.03]}$	11.60	9.09	1/4
lena	262944	$234543_{[12.10]}$	$226844_{[13.73]}$	$210786_{[19.83]}$	10.13	4.17	1/4
			$186359_{[29.13]}$	$175126_{[33.39]}$	25.33	20.00	1/3
peppers	262944	$232334_{[11.64]}$	$218175_{[17.03]}$	$199605_{[23.85]}$	14.09	6.25	1/4
			$180783_{[31.25]}$	$173561_{[33.99]}$	25.30	20.00	1/3

Table 2: Lossy compression of binary images.

file	file len	GZip len [%compr]	$Codec_2$ [%compr]	$Codec_1$ [%compr]	%Diff GZip	%loss	'.'/ char
ccitt7	513229	$109612_{[78.64]}$	$98076_{[80.89]}$	$91399_{[82.19]}$	16.62	16.67	1/5
			$93055_{[81.87]}$	$90873_{[82.29]}$	17.10	16.67	1/4
			$92658_{[81.95]}$	$85391_{[83.36]}$	22.10	25.00	1/3
test4	279213	$58736_{[78.96]}$	$57995_{[79.23]}$	$54651_{[80.42]}$	6.95	0,91	1/4
			$57714_{[79.32]}$	$54402_{[80.51]}$	7.38	1.27	1/3

5 LZW Encoding

Ziv and Lempel designed a class of compression methods based on the idea of back-reference: while the textfile is scanned, substrings or *phrases* are identified and stored in a *dictionary*, and whenever, later in the process, a phrase or concatenation of phrases is encountered again, this is compactly encoded by suitable pointers or indices [20, 30, 31]. In view of Theorem 1 and of the good

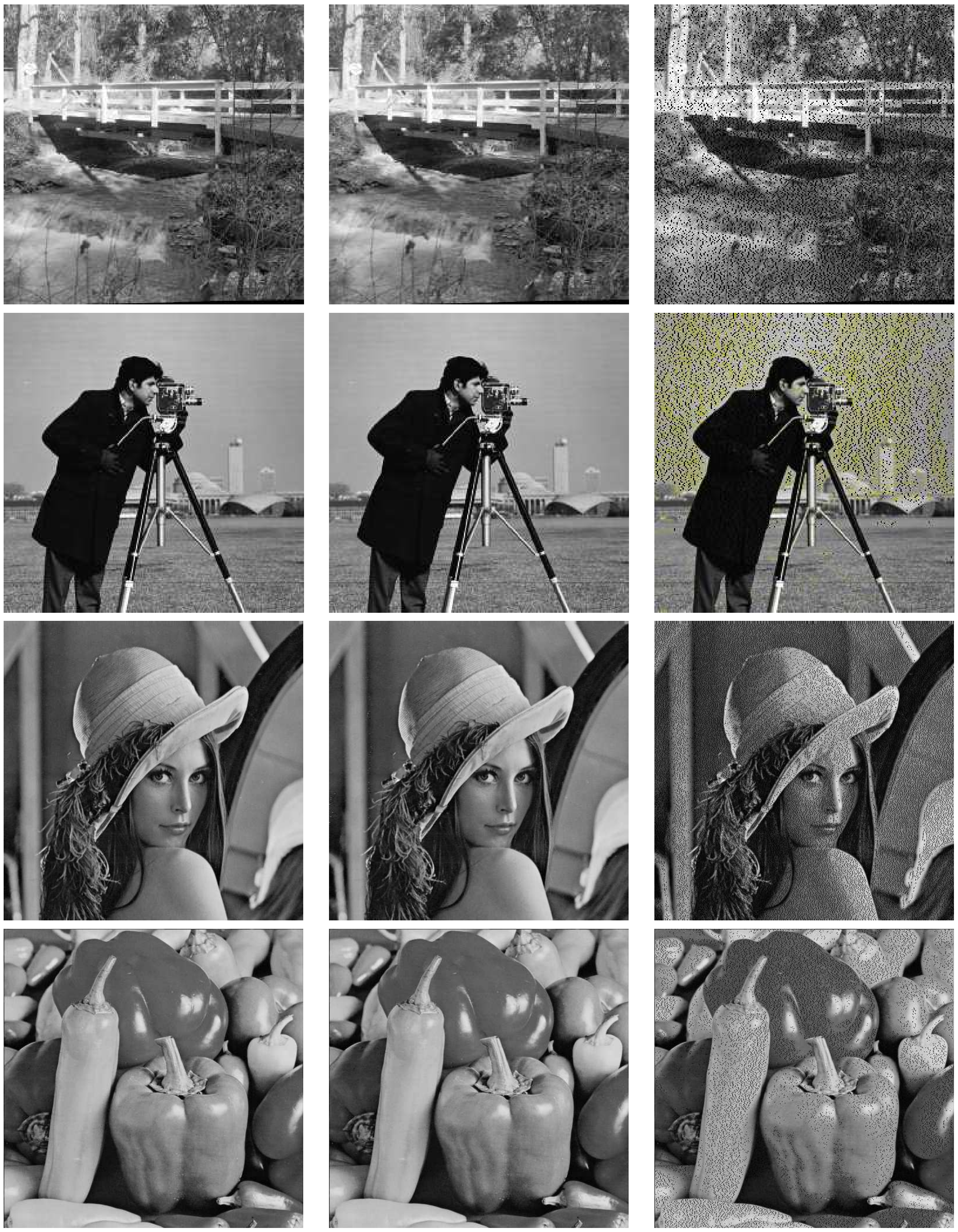


Figure 1: Compression and reconstruction of images. The original is on the first column, next to its reconstruction by interpolation of two closest solid pixels. Black dots used in the figures of the last column are used to display the distribution of the don't care characters. Compression of "Bridge" at 1/4 and 1/3 (shown here) '.'/char densities yields savings of 6.49% and 17.84% respectively. Correspondingly. 0,31% and 12,50% of the pixels differ from original after reconstruction. The lossy compression of Camera at 1/4 '.'/char density saves 11.60% over GZip. Only 6.67% of pixels differ from the original after reconstruction. Gains by "Lena" at 1/4 and 1/3 (shown) '.'/char density are respectively of 10,13% and 25,33%, while interpolation leaves resp. 3,85% and 10,13% differences from original. For "Peppers" (last row), the gains at 1/4 and 1/3 (shown) '.'/char densities were respectively 14,09% (5,56% the corresponding difference) and 25,30% (16,67% diff).

Table 3: Lossy compression of music (1 sample = 1 byte).

file	file len	GZip len [%compr]	*Codec*$_2$ [%compr]	*Codec*$_1$ [%compr]	%Diff GZip	%loss	'.'/ char
crowd	128900	$103834_{[19.44]}$	$92283_{[28.41]}$	$86340_{[33.01]}$	16.85	16.67	1/3
eclipse	196834	$171846_{[12.96]}$	$148880_{[24.36]}$	$139308_{[29.22]}$	18,93	9.09	1/4
			$114709_{[41.72]}$	$111058_{[43.57]}$	35.37	25.00	1/3

Table 4: Lossy vs. lossless performance.

file	file dim	GZip [%compr]	*Codec*$_1$ [%compr]	%loss	'.'/ char	Lossless [%compr]	%Diff GZip
bridge	66336	$61657_{[7.05]}$	$50656_{[23.63]}$	14.29	1/3	$59344_{[10.54]}$	3.75
camera	66336	$48750_{[26.51]}$	$43096_{[35.03]}$	9,09	1/4	$45756_{[31.02]}$	6.14
lena	262944	$234543_{[12.10]}$	$175126_{[33.39]}$	20.00	1/3	$199635_{[24.07]}$	14.88
peppers	262944	$232334_{[11.64]}$	$199605_{[23.85]}$	6.25	1/4	$211497_{[19.56]}$	8.97
			$173561_{[33.99]}$	20.00	1/3	$195744_{[25.55]}$	15.75
ccitt7	513229	$109612_{[78.64]}$	$90873_{[82.29]}$	16.67	1/4	$97757_{[80.09]}$	10.82
			$85391_{[83.36]}$	25.00	1/3	$89305_{[82.59]}$	18.53
test4	279213	$58736_{[78.96]}$	$54402_{[80.51]}$	1.27	1/3	$54875_{[80.34]}$	6.57
crowd	128900	$103834_{[19.44]}$	$86340_{[33.01]}$	16.67	1/3	$96903_{[24.82]}$	6.68
eclipse	196834	$171846_{[12.96]}$	$139308_{[29.22]}$	9.09	1/4	$159206_{[19.11]}$	7.36
			$111058_{[43.57]}$	25.00	1/3	$151584_{[22.98]}$	11.97

performance of motif based off-line compression [8], it is natural to inquire into the structure of ZL and ZLW parses which would use these patterns in lieu of exact strings. Possible schemes along these lines include, e.g., adaptations of those in [26], or more radical schemes in which the innovative add-on inherent to ZLW phrase growth is represented not by one symbol alone, but rather by that symbol plus the longest match with the substring that follows some previous occurrence of the phrase. In other words, the task of vocabulary build-up is assigned to the growth of (candidate), perhaps irredundant, 2-motifs.

Of the existing versions of the method, we recapture below the parse known as Ziv-Lempel-Welch, which is incarnated in the `compress` of UNIX. For the encoding, the dictionary is initialized with all the characters of the alphabet. At the generic iteration, we have just read a segment s of the portion of the text still to be encoded. With σ the symbol following this occurrence of s, we now proceed as follows: If $s\sigma$ is in the dictionary we read the next symbol, and repeat with segment $s\sigma$ instead of s. If, on the other hand, $s\sigma$ is not in the dictionary, then we append the dictionary index of s to the output file, and add $s\sigma$ to the dictionary; then reset s to σ and resume processing from the text symbol following σ. Once s is initialized to be the first symbol of the source text, "s belongs to the dictionary" is established as an invariant in the above loop. Note that the resulting set of phrases or codewords obeys the *prefix closure* property, in the sense that if a codeword is in the set, then so is also every one of its prefixes.

LZW is easily implemented in linear time using a trie data structure as the substrate [30, 31],

Table 5: Lossless compression of Calgary Corpus.

file	file len	GZip [%compr]	$Codec_1$ [%compr]	%loss	'.'/ char	Lossless [%compr]	%Diff GZip
bib	111261	$35063_{[68.49]}$	$36325_{[67.35]}$	3,70	1/3	$37491_{[66.30]}$	6.92
book1	768771	$313376_{[60.01]}$	$245856_{[68.01]}$	12.50	1/3	$277180_{[63.95]}$	11.55
book2	610856	$206687_{[66.16]}$	$197199_{[67.72]}$	4,35	1/4	$202713_{[66.81]}$	1.92
geo	102400	$68493_{[33.11]}$	$40027_{[60.91]}$	16.67	1/4	$63662_{[37.83]}$	7.05
news	377109	$144840_{[61.59]}$	$144541_{[61.67]}$	0.42	1/3	$144644_{[61.64]}$	0.14
obj1	21504	$10323_{[51.99]}$	$8386_{[61.00]}$	16.67	2/5	$9221_{[57.12]}$	10.68
obj2	246814	$81631_{[66.93]}$	$71123_{[71.18]}$	20.00	1/2	$83035_{[66.36]}$	-1.72
paper1	53161	$18577_{[65.06]}$	$19924_{[62.52]}$	1.75	1/3	$20174_{[62.05]}$	-8.60
paper2	82199	$29753_{[63.80]}$	$29920_{[63.60]}$	0.76	1/2	$30219_{[63.24]}$	-1.57
pic	513216	$56422_{[89.01]}$	$52229_{[89.82]}$	0.56	1/3	$52401_{[89.79]}$	7.13
progc	39611	$13275_{[66.49]}$	$13840_{[65.06]}$	1.32	1/2	$14140_{[64.30]}$	-6.52
progl	71646	$16273_{[77.29]}$	$17249_{[75.92]}$	0.58	1/3	$17355_{[75.78]}$	-6.65
progp	49379	$11246_{[77.23]}$	$12285_{[75.12]}$	0.64	1/3	$12427_{[74.83]}$	-10.50

and it requires space linear in the number of phrases at the outset. Another remarkable property of LZW is that the encoding and decoding are perfectly symmetrical, in particular, the dictionary is recovered while the decompression process runs (except for a special case that is easily taken care of).

We test the power of ZLW encoding on the motifs produced in greedy off-line schemata such as above. Despite the apparent superiority of such greedy off-line approaches in capturing long range repetitions, one drawback is in the encoding of references, which are bi-directional and thus inherently more expensive than those in ZLW. Our exercise consists thus of using the motifs selected in the greedy off-line to set up an initial vocabulary of motif phrases, but then encode these and their outgrowth while we carry out a parse of the source string similar to that of ZLW. Assuming that we have already selected the motifs to be used, this adaptation of ZLW to motifs requires to address primarily the following problems:

1. We need to modify the parsing in such a way that for every chosen motif, every one of its occurrences is used in the parsing.
2. We need to modify the dictionary in order to accommodate motifs in addition to strings. This is to be done while retaining *prefix closure*.

To ensure that a motif is correctly detected and deployed in the parsing, it has to be stored in the dictionary before its first occurrence is detected. This requires building a small dictionary that needs to be sent over to the decoder together with the encoded string. In order to enforce the prefix closure, all prefixes of a motif are inserted in the dictionary together with that motif.

With the dictionary in place, the parse phase of the algorithm proceeds in much the same way as in the original scheme, with the proviso that once a motif is chosen, then all of its occurrences are to be deployed. For this, the algorithm looks at each stage for the longest substring in the tree that does not interfere with the next motif occurrence already allocated from previous stages. The motif chosen in this way is then concatenated with the symbol following it and the result is inserted in the dictionary. In order to avoid the insertion of undesired don't cares in text regions

Table 6: Lossless compression of sequences from DNA yeast families.

file	file len	GZip [%compr]	$Codec_1$ [%compr]	%loss	'.'/ char	Lossless [%compr]	%Diff GZip
Spor EarlyII	25008	$8008_{[67.98]}$	$6990_{[72.05]}$	0.45	1/3	$7052_{[71.80]}$	11.94
Spor EarlyI	31039	$9862_{[68.23]}$	$8845_{[71.50]}$	0.36	1/3	$8914_{[71.28]}$	9.61
Helden CGN	32871	$10379_{[68.43]}$	$8582_{[73.89]}$	1.33	1/3	$8828_{[73.14]}$	14.94
Spor Middle	54325	$16395_{[69.82]}$	$14839_{[72.68]}$	0.36	1/4	$14924_{[72.53]}$	8.97
Helden All	112507	$33829_{[69.93]}$	$29471_{[73.81]}$	1.56	1/4	$29862_{[73.46]}$	11.73
Spor All	222453	$68136_{[69.37]}$	$56323_{[74.68]}$	1.61	1/3	$57155_{[74.31]}$	16.12
All Up 400k	399615	$115023_{[71.22]}$	$93336_{[76.64]}$	14.29	1/3	$106909_{[73.25]}$	7.05

Table 7: Synopsis of compression rates for sequences in the yeast DNA by various lossless methods. The figure in parenthesis is the percentage gain of $Codec_1$ versus other methods.

File	File Len	Huffman Pack [%diff]	LZ-78 Compress [%diff]	LZ-77 GZip [%diff]	BWT BZip [%diff]	$Codec_1$ Lossless
Spor EarlyII	25008	$7996_{[13.4]}$	$7875_{[11.7]}$	$8008_{[13.6]}$	$7300_{[3.5]}$	7052
Spor EarlyI	31039	$9937_{[11.5]}$	$9646_{[8.2]}$	$9862_{[10.6]}$	$9045_{[1.5]}$	8914
Helden CGN	32871	$10590_{[20.0]}$	$10223_{[15.8]}$	$10379_{[17.6]}$	$9530_{[8.0]}$	8828
Spor Middle	54325	$17295_{[15.9]}$	$16395_{[9.9]}$	$16395_{[9.9]}$	$15490_{[3.8]}$	14924
Helden All	112507	$36172_{[21.1]}$	$33440_{[12.0]}$	$33829_{[13.3]}$	$31793_{[6.5]}$	29862
Spor All	222453	$70755_{[23.8]}$	$63939_{[11.9]}$	$68136_{[19.2]}$	$61674_{[7.9]}$	57155
All Up 400k	399615	$121700_{[13.8]}$	$115029_{[7.6]}$	$115023_{[7.6]}$	$112363_{[5.1]}$	106909

not encoded by motifs, that symbol is treated as mismatching all other characters at this stage of the search.

Decoding is easier. The recovery follows closely the standard ZLW, except for initialization of the dictionary. The only difference is thus that now the decoder receives, as part of the encoding, also an initial dictionary containing all motifs utilized, which are used to initialize the trie.

The tables below summarize results obtained on gray-scale images (Table 9, 1 pixel = 1 byte), the Calgary Corpus (Table 10), and genetic data (Table 11). For each case, the compression is reported first for lossy encoding with various don't care densities, then also for the respective lossless completions.

Conclusion

Irredundant motifs seem to provide an excellent repertoire of codewords for grammar based compression and syntactic inference of documents of various kinds. Various completion strategies and possible extensions (e.g., to nested descriptors) and generalizations (notably, to higher dimensions) suggest that the notions explored here can develop in a versatile arsenal of data compression methods capable of bridging lossless and lossy textual substitution in a way that is both aesthetically pleasant and practically advantageous. Algorithms for efficient motif extraction as well as for their

Table 8: Compression, fidelity and loss in reconstruction of grey scale images.

File	File len	GZip len [%compr]	*Codec*$_1$ [%compr]	Diff % GZip	%Loss	’.’/ car	%Loss per pixel over recon pix	%Loss per pixel over all pix
bridge	66336	$61657_{[7.05]}$	$57655_{[13.08]}$	6.49	0.42	1/4	5.67	0.02
			$50656_{[23.63]}$	17.84	14.29	1/3	7.69	0.90
camera	66336	$48750_{[26.51]}$	$43090_{[35.03]}$	11.60	9.09	1/4	0.78	0.05
lena	262944	$234543_{[12.10]}$	$210786_{[19.83]}$	10.13	4.17	1/4	7.26	0.27
			$175126_{[33.39]}$	25.33	20.00	1/3	5.11	0.81
peppers	262944	$232334_{[11.64]}$	$199605_{[23.85]}$	14.09	6.25	1/4	1.53	0.08
			$173561_{[33.99]}$	25.30	20.00	1/3	3.29	0.52

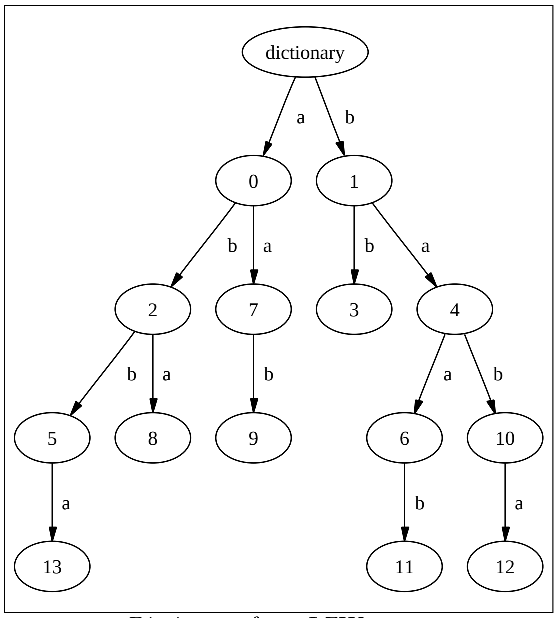
Table 9: Lossy/Lossless compression of gray-scale images using LZW-like encoding.

File	File len	GZip len	LZW-like lossy	% Diff GZip	% Loss	LZW-like lossless	% Diff GZip	‘.’/ car
bridge	66.336	61.657	38.562	37.46	0.29	38.715	37.21	1/4
			38.366	37.78	5.35	42.288	31.41	1/3
camera	66.336	48.750	34.321	29.60	0.00	34.321	29.60	1/6
			34.321	29.60	0.06	34.321	29.60	1/5
			32.887	32.54	6.16	35.179	27.84	1/4
lena	262.944	234.543	120.308	48.71	1.36	123.278	47.44	1/4
			123.182	47.48	7.32	135.306	42.31	1/3
peppers	262.944	232.334	117.958	49.23	1.75	121.398	47.75	1/4
			119.257	48.67	4.45	129.012	44.47	1/3

efficient deployment in compression are highly desirable from this perspective. In particular, algorithms for computing the statistics for maximal sets of *non-overlapping* occurrences for each motif should be set up for use in gain estimations, along the lines of the constructions given in [10] for solid motifs. Progress in these directions seems not out of reach.

Acknowledgments

The authors are indebted to R. Ahlswede for the warm hospitality and congenial atmosphere provided at the ZiF Meetings in Bielefeld. D. Epstein and, independently, S. Lonardi brought reference [19] to the attention of one of the authors. The anonymous referees of [8, 9] gave useful comments on those preliminary versions of parts of this paper.



Dictionary from LZW parse

0	1	1	2	4	0	2	7	4	6	10	5	7
a	b	b	ab	ba	a	ab	aa	ba	baa	bab	abb	aa

Figure 2: Illustrating ZLW encoding and parse (top) as applied to the string *abbabbaaabaababaababababbaa*. .

References

[1] M.J. Atallah, Y. Genin and W. Szpankowski, "Pattern Matching Image Compression: Algorithmic and Empirical Results", IEEE Transactions on PAMI **21**:7, pp. 614–629 (1999).

[2] A. Apostolico, "On the Efficiency of Syntactic Feature Extraction and Approximate Schemes for Data Compression", *Proceedings of the 5th International Conference on Pattern Recognition.*, pp. 982–984, Miami, Florida (1980).

[3] A. Apostolico and M.J. Atallah, "Compact Recognizers of Episode Sequences", *Information and Computation*, **174**, pp. 180-192 (2002).

[4] A. Apostolico and Z. Galil (Eds.), *Pattern Matching Algorithms*, Oxford University Press, New York (1997).

[5] A. Apostolico and A. Fraenkel, "Robust Transmission of Unbounded Strings Using Fibonacci Representations", *IEEE Transactions on Information Theory*, vol. 33, no. 2, pp. 238–245 (1987).

[6] A. Apostolico and S. Lonardi, "Off-line Compression by Greedy Textual Substitution", *Proceedings of the IEEE* , **88**(11): 1733–1744 (2000).

[7] A. Apostolico and L. Parida, "Incremental Paradigms of Motif Discovery", *Journal of Computational Biology*, **11**(1), pp. 15-25 (2004) .

[8] A. Apostolico and L. Parida, "Compression and the Wheel of Fortune" Proceedings of *IEEE DCC Data Compression Conference*, pp. 143–152 Computer Society Press, (2003).

[9] A. Apostolico, M. Comin and L. Parida, "Motifs in Ziv-Lempel-Welch Clef" Proceedings of *IEEE DCC Data Compression Conference*, pp. 72–81 Computer Society Press, (2004).

[10] A. Apostolico and F.P. Preparata, "Data Structures and Algorithms for the String Statistics Problem", *Algorithmica*, **15**, pp. 481–494 (1996).

Table 10: Lossy/Lossless compression of the Calgary Corpus by LZW-like encoding.

File	File len	GZip len	LZW-like lossy	% Diff GZip	% Loss	LZW-like lossless	% Diff GZip	'.'/ car
bib	111.261	35.063	26.679	23.91	14.11	34.174	2.54	1/4
			26.679	23.91	14.11	34.174	2.54	1/3
geo	102.400	68.493	30.951	54.81	19.43	57.098	16.64	1/4
			33.334	51.33	20.63	58.038	15.26	1/3
news	377.109	144.840	104.807	27.64	11.42	128.429	11.33	1/4
			106.483	26.48	19.95	153.243	-5.80	1/3
obj1	21.504	10.323	8.447	18.17	8.38	9.642	6.60	1/4
			7.409	28.23	18.03	9.849	4.59	1/3
			6.736	34.75	21.61	8.521	17.46	2/5
obj2	246.814	81.631	56.810	30.41	12.75	67.857	16.87	1/3
			53.094	34.96	19.88	67.117	17.78	1/2
paper1	53.161	18.577	16.047	13.62	13.63	19.411	-4.49	1/3
			15.626	15.89	18.22	19.198	-3.34	1/2
paper2	82.199	29.753	23.736	20.22	15.19	28.743	3.39	1/3
			22.519	24.31	36.44	35.390	-18.95	1/2
pic	513.216	56.422	36.491	35.32	0.56	36.599	35.13	1/4
progc	39.611	13.275	11.576	12.80	6.88	12.812	3.49	1/4
			11.381	14.27	8.69	12.854	3.17	1/3
			11.010	17.06	24.86	15.246	14.85	1/2
progl	71.646	16.273	14.828	8.88	2.72	15.601	4.13	1/4
			14.748	9.37	6.99	16.149	0.76	1/3
			14.676	9.81	7.70	16.261	0.07	2/5
progp	49.379	11.246	10.287	8.53	3.83	10.879	3.26	1/4
			10.265	8.72	7.43	11.328	-0.73	1/3
			10.416	7.38	7.12	11.569	-2.87	2/5

[11] T. Berger, *Rate Distortion Theory: A Mathematical Basis for Data Compression*, Prentice Hall, Englewood Cliffs, N.J. (1971).

[12] T. Berger, J. D. Gibson, *Lossy Source Coding.* IEEE Transactions on Information Theory **44**(6): 2693-2723 (1998).

[13] A. Blumer, J. Blumer, A. Ehrenfeucht, D. Haussler, M.T. Chen and J. Seiferas, "The Smallest Automaton Recognizing the Subwords of a Text", *Theoretical Computer Science*, **40**, 31-55 (1985).

[14] S. DeAgostino and J. A. Storer, "On-line Versus Off-line Computation in Dynamic Text Compression", *Inform. Process. Lett.*, 59(3):169–174 (1996).

[15] K. S. Fu and T. L. Booth, "Grammatical Inference: Introduction and Survey – Part I", *IEEE Transactions on Systems, Man and Cybernetics*, 5:95–111 (1975).

[16] K. S. Fu and T. L. Booth, "Grammatical Inference: Introduction and Survey – Part II", *IEEE Transactions on Systems, Man and Cybernetics*, 5:112–127 (1975).

[17] J.C. Kieffer, "A Survey of the Theory of Source Coding with a Fidelity Criterion". *IEEE Transactions on Information Theory*, 39(5):1473–1490 (1993).

[18] I. Kontoyiannis, "An Implementable Lossy Version of Lempel-Ziv Algorithm –Part 1: Optimality for Memoryless Sources", *IEEE Transactions on Information Theory*, 45:2293–2305 (1999).

Table 11: Lossy/Lossless compression of sequences from DNA yeast families by LZW-like encoding.

File	File len	GZip len	LZW-like lossy	% Diff GZip	% Loss	LZW-like lossless	% Diff GZip	'.'/ car
Spor EarlyII	25.008	8.008	6.137	23.36	16.93	7.430	7.22	1/4
			6.163	23.04	12.01	7.052	11.94	1/3
Spor EarlyI	31.039	9.862	7.494	24.01	14.66	8.865	10.11	1/4
			7.494	24.01	14.66	8.865	10.11	1/3
Helden CGN	32.871	10.379	7.728	25.54	15.36	9.330	10.11	1/3
Spor Middle	54.325	16.395	11.565	29.46	0.38	11.672	28.81	1/4
			11.555	29.52	0.43	11.703	28.62	1/3
Helden All	112.507	33.829	25.873	23.52	18.83	32.029	5.32	1/4
			26.010	23.11	18.86	32.182	4.87	1/3
Spor All	222.453	68.136	48.035	29.50	18.98	60.042	11.88	1/4
			47.896	29.71	19.00	59.955	12.01	1/3
All Up 400k	399.615	115.023	90.120	21.65	18.62	110.659	3.79	1/3

[19] E. Lehman and A. Shelat, "Approximation Algorithms for Grammar Based Compression", *Proceedings of the eleventh ACM-SIAM Symposium on Discrete Algorithms (SODA 2002)*, January 2002.

[20] A. Lempel and J. Ziv, "On the complexity of finite sequences," *IEEE Trans. on Inform. Theory*, vol. 22, pp. 75–81, 1976.

[21] T. Luczak andW. Szpankowski, "A Suboptimal Lossy Data Compression Algorithm Based on Approximate Pattern Matching", *IEEE Transactions on Information Theory*, 43(5):1439–1451 (1997).

[22] C. Neville-Manning, I. H. Witten, and D. Maulsby, "Compression by Induction of Hierarchical Grammars", In *DCC: Data Compression Conference*, pages 244–253. IEEE Computer Society TCC (1994).

[23] L. Parida, I. Rigoutsos, D. Platt, An Output-sensitive Flexible Pattern Discovery Algorithm, *Combinatorial Pattern Matching (CPM 2001)*, (A. Amir and G. Landau, Eds.), LNCS vol **2089**, pp 131–142, (2001).

[24] L. Parida, I. Rigoutsos, A. Floratos, D. Platt, Y. Gao, "Pattern Discovery on Character Sets and Real-valued Data: Linear Bound on Irredundant Motifs and Polynomial Time Algorithms", *Proceedings of the eleventh ACM-SIAM Symposium on Discrete Algorithms (SODA 2000)*, January 2000, pp 297–308, (2000).

[25] I. Rigoutsos, A. Floratos, L. Parida, Y. Gao, D. Platt, "The Emergence of Pattern Discovery Techniques in Computational Biology", *Journal of Metabolic Engineering*, **2**(3):159-177, (2000).

[26] I. Sadeh "On Approximate String Matching", *Proceedings of DCC 1993*, IEEE COmputer Society Press, 148–157 (1993).

[27] J. A. Storer, *Data Compression: Methods and Theory*, Computer Science Press (1988).

[28] J. Wang, B. Shapiro, and D. Shasha (Eds.), *Pattern Discovery in Biomolecular Data: Tools, Techniques, and Applications* Oxford University Press (1999).

[29] M. Waterman, *Introduction to Computational Biology*, Chapman and Hall (1995).

[30] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Trans. on Inform. Theory*, vol. IT-23, no. 3, pp. 337-343, (1977).

[31] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding," *IEEE Trans. on Inform. Theory*, vol. 24, no. 5, pp 530-536 (1978).

APPENDIX

A Proof of Theorem 1.

Let m be a 2-motif in $\mathcal{B}$, and $\mathcal{L}_m = (l_1, l_2, \ldots, l_p)$ be its occurrence list. The claim is true for $p = 2$. Indeed, let $i = l_1$ and $j = l_2$, and consider the meet m' of suf_i and suf_j. By the maximality in composition of m, we have that $m' \preceq m$. On the other hand, for any motif $\hat{m}$ with occurrences at i and j it must be $\hat{m} \preceq m'$, whence, in particular, $m \preceq m'$. Thus, $m = m'$. Assume now $p \geq 3$ and that there is no pair of indices i and j in $\mathcal{L}_m$ such that m is the meet of suf_i and suf_j. Again, for any choice of i and j in $\mathcal{L}_m$, we must have that $m \preceq m'$, where m' denotes as before the meet of suf_i and suf_j. Therefore, we have that $m \preceq m'$ but $m \neq m'$ for all choices of i and j. Assume now one such choice is made. By the maximality of m, it cannot be that m' is the meet of all suffixes with beginning in $\mathcal{L}_m$. Therefore, there must be at least one index k such that m' differs either from the meet of suf_k and suf_i or from the meet of suf_k and suf_j, or from both. Let, to fix the ideas, m'' be this second meet. Since $m \preceq m''$ and $m \preceq m'$ then $\mathcal{L}_{m'}$ and $\mathcal{L}_{m''}$ are sublists of $\mathcal{L}_m$, by Lemma 1. In other words, $\mathcal{L}_m$ can be decomposed into two or more lists of maximal motifs such that their union implies m and its occurrences. But this contradicts the assumption that m is irredundant. ∎

B The Set Union Problem, SUP(n, q).

Given n sets $S_1, S_2 \ldots, S_n$ on q elements each, find all the sets S_i such that $S_i = S_{i_1} \cup S_{i_2} \cup \ldots \cup S_{i_p}$ $i \neq i_j$, $1 \leq j \leq p$.

This is a very straightforward algorithm (this contributes an additive term to the overall complexity of the pattern detection algorithm): For each set S_i, we first obtain the sets S_j $j \neq i, j = 1 \ldots n$ such that $S_j \subset S_i$. This can be done in $O(nq)$ time (for each i). Next, we check if $\cup_j S_j = S_i$. Again this can be done in $O(nq)$ time. Hence the total time taken is $O(n^2 q)$.