Literal Shuffle of Compressed Words

Alberto Bertoni[1], Christian Choffrut[2], and Roberto Radicioni[1]

[1] Dip. di Scienze dell'Informazione, Università degli Studi di Milano, Via Comelico 39/41, 20135 Milano - Italy
`{bertoni,radicioni}@dsi.unimi.it`

[2] L.I.A.F.A. (Laboratoire d'Informatique Algorithmique, Fondements et Applications), Université Paris VII, 2 Place Jussieu, 75221 Paris - France
`Christian.Choffrut@liafa.jussieu.fr`

Abstract. Straight-Line Programs (SLP) are widely used compressed representations of words. In this work we study the rational transformations and the literal shuffle of words compressed via SLP, proving that the first preserves the compression rate, while the second does not. As a consequence, we prove a tight bound for the descriptional complexity of 2D texts compressed via SLP. Finally, we observe that the Pattern Matching Problem for texts expressed by the literal shuffle of compressed words is NP-complete. However, we present a parameter-tractable algorithm for this problem, working in polynomial time whenever the length of the pattern is polynomially related to that of the text.

1 Introduction

Straight-line programs (SLP) are a widely accepted representation of compressed texts (see, for instance, [16, 13, 12, 11]). A SLP is a grammar in Chomsky Normal Form generating only one word; the grammar can be seen as a compressed representation of the word. Such a representation suggests a natural measure of descriptional complexity for a word, consisting of the SLP of smallest size that generates it. The compression rate of SLPs is comparable to that of Lempel-Ziv factorization. Indeed, given the LZ-encoding of a word, it is possible to obtain a SLP of the same compressed size, up to a log factor, that generates the same word ([17]).

Since the output size of these compression techniques could be logarithmic with respect to the length of the generated word, it is useful to design algorithms for problems on compressed texts without full unpacking. Generally, in this context, grammar compression is more convenient than LZ-factorization. For some problems, such as Equality and Pattern Matching with grammar compressed words as input, polynomial time algorithms have been found ([15, 9]); for other problems, the compressed version becomes NP-hard (for instance, computing Hamming distance, as proved in [9]).

In this work, we consider some operations on strings and study the problem of implementing such operations in compressed representations. In particular, we consider the rational transformations and the literal shuffle. The literal shuffle consists of merging two words of equal length from left to right alternating exactly one symbol of the first word and one of the second (for example the literal shuffle of “lug” and “one” is “lounge”). The “inverse operation” R (L) consists of selecting the subword composed by the symbols in odd (even, respectively) position in the input word.

These operations play an important role in Cooley-Tukey Algorithm for the fast computation of the Discrete Fourier Transform [3]. This technique is based on a Divide and Conquer strategy which recursively breaks up a string by using R and L operations, while the merging phase consists of applying the literal shuffle to the partial solutions. A natural question is whether it is possible to apply this technique in a compressed context, that is, to execute efficiently R, L and the literal shuffle on grammar compressed strings.

First of all, we prove that rational transformations preserve the compression rate, while, in general, this does not hold for literal shuffle. This fact is proved by exploiting a construction that relates the circuital complexity of boolean functions with the descriptional complexity defined in terms of SLPs.

This result is then applied to compressed pictures. 2D-texts can be compressed by using a 2D version of SLPs. The structure of 2D SLPs is more complex than that of SLPs. Indeed, it is known that, while factors of logarithmically compressible words are still logarithmically compressible, this does not hold for 2D texts. In particular, there exists an infinite number of logarithmically compressible pictures having at least one section (row or column) not logarithmically compressible ([2]). We obtain a bound for the descriptional complexity of the sections of a compressed picture which depends on their position in the picture. Such a bound is proved to be tight, in some sense.

Finally, we study the problem of deciding whether a word is a factor of a text, where both the word and the text are represented by the literal shuffle of compressed words given as input. We prove that the problem is NP-complete also if the word to be searched for is 11. However, we present an algorithm working in polynomial time whenever the length of the pattern is polynomially related to that of the text.

2 Preliminaries

Given a word $w \in \Sigma^*$, we denote by $w[i]$ the i-th symbol of w and by $w[i,j]$ the factor $w[i] \cdots w[j]$ of w, where $1 \leq i \leq j \leq |w|$. We call $\mathrm{Fact}\,(w)$ the set of the factors of w.

For the sake of simplicity, in the following we consider $\Sigma = \{0,1\}$; given a word $x \in \{0,1\}^n$, $b(x)$ is the base-2 integer whose binary representation is x and, with an abuse of notation, we intend $\underline{x}$ as the vector of components $b(x[1]), \ldots, b(x[n])$. By $\underline{0}$ ($\underline{1}$), we denote a vector whose components are all 0 (1, respectively); its dimension is specified only in the case of ambiguity. Given two vectors $\underline{a} = (a_1, \ldots, a_n)$ and $\underline{b} = (b_1, \ldots, b_m)$, we denote by $\underline{a} \odot \underline{b}$ the concatenation $(a_1, \ldots, a_n, b_1, \ldots, b_m)$ and, if $n = m$, by $\underline{a} \cdot \underline{b}$ the sum $\sum_i a_i b_i$.

2.1 Straight-Line Programs

A *straight-line program* (*SLP*) is a sequence of labelled instructions of the form

$$X_1 = 0, \quad X_2 = 1, \quad X_k \ = \ X_i X_j \qquad 0 < i, j < k, \quad k = 3, \ldots, n.$$

The output of a SLP Φ is the word generated by performing all the concatenations from X_3 to X_n and is denoted by eval(Φ), while we write $\text{eval}_\Phi(X_k)$ for the word obtained by performing the first k concatenations in Φ. The number n of instructions in Φ is called its *size* and is denoted by $|\Phi|$. For every $w \in \{0,1\}^*$, as *descriptional complexity* of w we consider the size $g(w)$ of the smallest SLP generating w.

Since the computational complexity of a word can be logarithmic with respect of its size, many classical problems on words are studied in their compressed version, that is, considering SLPs as input instead of words.

For instance, the input of the compressed version of Equality is a pair (Φ, Ψ) of straight-line programs and the question is to decide whether eval(Φ) = eval(Ψ). Analogously, the question of the compressed version of Pattern Matching is to decide whether eval(Ψ) is a factor of eval(Φ). The first result in this direction is in [15] where a polynomial time algorithm for Equality is shown, while the best algorithm for Compressed Pattern Matching is presented in [9].

2.2 Lempel-Ziv Factorization

The *LZ-factorization* of a word w is a decomposition $f_1 \cdots f_k = w$, where $f_1 = w[1]$ and f_{i+1} is the shortest factor not appearing in $f_1 \cdots f_i$. We call *LZ-factors* of w the factors appearing in its LZ-factorization. The LZ-encoding of w is the sequence $LZ(w) = (f_1, \ldots, f_k)$, where every LZ-factor $f_i = w[a, b]$ is exclusively expressed by a and b. LZ-encoding gives a very efficient lossless compression technique, used in several compression standards ([8, 7]).

The size of $LZ(w)$ is the number of its LZ-factors and is denoted by $|LZ(w)|$. In [17] it is shown that $g(w) \geq |LZ(w)|$ and $g(w) = O(|LZ(w)| \times \log |w|)$ for every w. Moreover, we give a simple lower bound for the size of a LZ-factorization:

Lemma 1. *For every $w \in \{0,1\}^*$, $|LZ(w)| \geq |Fact(w) \cap 10^*1|$.*

Proof. The LZ-factorization of w contains a LZ-factor for each first occurrence of 10^t1 with different t. Indeed, if we scan w from left to right and run into a factor 10^t1 for the first time, then two cases are possible: either we already ran into a sequence of zeros of length $s > t$, and then a new LZ-factor necessarily starts immediately after 10^t1 in w, or the longest sequence of zeros has length $s < t$, then a new LZ-factor starts from the $(s+1)$th zero of 10^t1. □

3 Rational Transformations and SLPs

In this section we study rational transformations of compressed words. First, we recall the notion of deterministic rational transducer, defining from word to word rational transformations. Then we prove that rational transformations on compressed words preserve the compression rate.

A *deterministic rational transducer* is a 5-tuple $A = (\Sigma, \Gamma, Q, q_0, \delta)$, where Σ is the input alphabet, Γ is the output alphabet, Q is the set of states, $q_0 \in Q$ is the initial state and $\delta = (\delta_Q, \delta_\Gamma)$ is the transition function, with $\delta_Q : Q \times \Sigma \to Q$ and $\delta_\Gamma : Q \times \Sigma \to \Gamma^*$. We denote $(\delta_Q(q,\sigma), \delta_\Gamma(q,\sigma))$ as $\delta(q,\sigma)$.

The extension of δ_Q to Σ^* is similar to the case of finite state automata, we set $\delta_Q^*(q,\epsilon) = q$ and $\delta_Q^*(q, w\sigma) = \delta_Q(\delta_Q^*(q,w),\sigma)$ for every $w \in \Sigma^*$. The extension of δ_Γ is different: we set $\delta_\Gamma^*(q,\epsilon) = \epsilon$, and

$$\delta_\Gamma^*(q, w\sigma) = \delta_\Gamma^*(q,w)\delta_\Gamma(\delta_Q^*(q,w),\sigma),$$

where $w \in \Sigma^*$. The rational transformation applied by A to a word $w \in \Sigma^*$ is the word $A(w) = \delta_\Gamma^*(q_0, w)$.

To our aim, we consider the set $S = \{w \mid \delta_\Gamma(q,\sigma) = w, q \in Q, \sigma \in \Sigma\}$ and define the size of A as $|A| = |Q| + \sum_{w \in S} |w|$. Hence, in the context of compressed words, we introduce the following problem:

Problem: *Compressed Rational Transformation (CRT)*
Instance: A deterministic rational transducer A and a SLP Φ;
Question: A SLP generating $A(\mathrm{eval}(\Phi))$.

The CRT problem can be solved in polynomial time, as stated in the following

Theorem 1. *Given a rational transducer A and a SLP Φ, there is a $O(|A| \times |\Phi|)$ algorithm for the CRT problem with input A and Φ.*

Proof. Let $A = (\Sigma, \Gamma, Q, q_0, \delta)$ and $n = |\Phi|$. We first compute a table T with entries (X_k, q) with $X_k \in \Phi$ and $q \in Q$, such that $T(X_k, q) = \delta_Q^*(q, \mathrm{eval}_\Phi(X_k))$. The table T can be computed in time $O(|Q| \times |\Phi|)$ by giving an order to the pairs (X_k, q_i) which preserves the order in Φ and then, for every instruction $X_k = X_i X_j$ and every $q \in Q$, computing the entry $T(X_k, q)$ as $T(X_j, T(X_i, q))$. If $X_k = \sigma$, then $T(X_k, q) = \delta_Q(q, \sigma)$.

We obtain a new SLP Ψ by translating each variable X_k of Φ in $|Q|$ variables $X(k, q)$ of the form

$$X(k,q) \;=\; \begin{cases} X(i,q)\; X(j,T(X_i,q)) & \text{if } \; X_k = X_iX_j, \\ \Psi(q,\sigma) & \text{if } \; X_k = \sigma, \;\; \sigma \in \Sigma, \end{cases}$$

where $\Psi(q,\sigma)$ is a SLP such that $\mathrm{eval}(\Psi(q,\sigma)) = \delta_\Gamma(q,\sigma)$. By setting $X(n,q_0)$ as the last variable of Ψ, we have $\mathrm{eval}_\Psi(X(n,q_0)) = A(\mathrm{eval}(\Phi))$ by construction. Every $\Psi(q,\sigma)$ has size at most $|\delta_\Gamma(q,\sigma)|$, hence $|\Psi| = O(|A| \times |\Phi|)$. $\square$

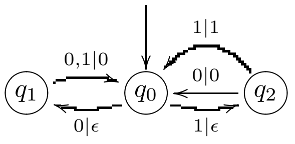
A straightforward consequence of Theorem 1 is that the compression properties of SLPs are preserved by rational transformations.

Corollary 1. *Let A be a fixed rational transducer. Then, $g(A(w)) = O(g(w))$ for every $w \in \Sigma^*$.*

Example 1. Consider the following rational transducer

$$A \;=\; (\{0,1\},\{0,1\},\{q_0,q_1,q_2\},q_0,\delta),$$

where $\delta(q_0,0) = (q_1,\epsilon)$, $\delta(q_1,0) = \delta(q_1,1) = \delta(q_2,0) = (q_0,0)$, $\delta(q_0,1) = (q_2,\epsilon)$ and $\delta(q_2,1) = (q_0,1)$.



Such a transducer reads the symbols of a word in $\{0,1\}^*$ two by two, and writes 1 for 11 and 0 for 00, 01 and 10.

Let $\Phi = (X_1 = 0, X_1 = 1, [X_k = X_{k-1}X_{k-2}]_{k\in[2..6]})$ be the SLP that generates the 6th Fibonacci word $\mathrm{eval}(\Phi) = 10110101$. Applying the algorithm of Th. 1, we obtain the (opportunely simplified) straight-line program $\Psi = (X_1 = 0, X_2 = 1, X_3 = X_1X_1, X_4 = X_1X_2, X_5 = X_4X_3)$, which generates the word 0100.

4 Lohrey Strings

In this section we recall a construction due to Lohrey ([10]), useful for study the computational complexity of some compressed word problems. The SubsetSum problem consists in deciding, given as input a vector $\underline{w}$ of integers and a target integer t, if there is at least one selection of entries in $\underline{w}$ whose sum is t. It can be formally defined as

Problem: *Subset Sum (*SubsetSum*)*
Instance: $\underline{w} \in \mathbf{N}^n$, $t \in \mathbf{N}$;
Question: does there exist $\underline{x} \in \{0,1\}^n$ such that $\underline{x} \cdot \underline{w} = t$?

It is a well known NP-complete problem and its counting version, consisting in defining the cardinality of the set $\{\underline{x} \in \{0,1\}^n \mid \underline{x} \cdot \underline{w} = t\}$, is $\sharp P$-complete. In the context of straight-line programs, it has been used to prove that computing the Hamming distance of two compressed words is a $\sharp P$-complete problem ([9]). The proof makes use of the so called Lohrey strings [10], couples of words representing instances of SubsetSum problem that have an exponential compression rate.

Let $I = (\underline{w}, t)$ be an instance of SubsetSum, with $\underline{w} \in \mathbf{N}^n$ and define $s = \underline{1} \cdot \underline{w}$. The *Lohrey strings* of I are the two words

$$\xi(I) \;=\; (0^t 1 0^{s-t})^{2^n} \qquad \xi'(I) \;=\; \prod_{\substack{x\in\{0,1\}^n \\ b(x)=0..2^n-1}} (0^{\underline{x}\cdot\underline{w}} 1 0^{s-\underline{x}\cdot\underline{w}})$$

of length $(s+1)2^n$. Informally, $\xi(I)$ encodes t by 2^n blocks of length $s+1$ made of zeros in all places except in the $(t+1)$-th. On the other hand, $\xi'(I)$ encodes the sums of all the possible subsets of $\underline{w}$ by setting to 1 the only bit in position $\underline{x} \cdot \underline{w}$ in the x-th block, for every $x \in \{0,1\}^n$.

The relevance of Lohrey strings is depicted by the following

Lemma 2. *Let $I = (\underline{w}, t)$ be an instance of* SubsetSum *with $\underline{w} \in \mathbf{N}^n$. Then, $g(\xi(I)), g(\xi'(I)) = n^{O(1)}$.*

Proof. This lemma is a special case of Theorem 6 in [10]. □

5 Literal Shuffle of Compressed Words

In this section we consider the operations of bitwise AND and literal shuffle between words. Let $x, y \in \{0,1\}^n$, with $n > 0$; the bitwise AND $x \wedge y$ is $(x \wedge y)[i] = x[i] \wedge y[i]$ for $i = 1, \ldots, n$, while the *literal shuffle* ([1]) of x and y is defined as

$$x \sqcup\!\sqcup y \;=\; x[1]y[1]x[2]y[2] \cdots x[n]y[n],$$

Its "inverse operations" are L and R, where, for a word $w \in \{0,1\}^{2n}$,

$$L(w) \;=\; w[1]w[3] \cdots w[2n-1], \qquad R(w) \;=\; w[2]w[4] \cdots w[2n].$$

Operations L, R and $\sqcup\!\sqcup$ play an important role in many algorithms (such as Fast Fourier Transform for analysis and compression of digital signals) and it would be interesting to work using these operations in a compressed representation. L and R preserve the compression rate, since it is easy to construct the deterministic rational transducers implementing such operations. Unfortunately, this does not hold for the literal shuffle, as proved in this section.

First of all, we prove a technical lemma that allows to transform constructions using boolean circuits into constructions using SLPs.

Lemma 3. *Let C be a circuit computing the boolean function $f(\underline{x})$ in the variables $x_1, \ldots, x_n$. Then, there exist two SLP Φ and Ψ such that*

– $|\Phi|, |\Psi| = |C|^{O(1)}$;
– $|eval(\Phi)| = |eval(\Psi)| = 2^{n+m}$, *with* $m = n^{O(1)}$;
– $f(\underline{x}) = 1 \implies \exists! z \in \{0,1\}^m \mid eval(\Phi)[b(xz)] = eval(\Psi)[b(xz)] = 1$;
– $f(\underline{x}) = 0 \implies \forall z \in \{0,1\}^m \mid eval(\Phi)[b(xz)] \wedge eval(\Psi)[b(xz)] = 0$;

Proof. Without loss of generality, suppose C is built using NAND gates. Then, it is easy to construct a 3-CNF formula ϕ for f with $O(|C| + n)$ variables and clauses by adding the boolean variables $y_1, \ldots, y_{|C|}$ to the initial $x_1, \ldots, x_n$. Let y_k represent the output of a gate k in C and let a and b be its inputs. Then, ϕ contains the clauses defining $y_k = \overline{a \wedge b}$. Moreover, it contains further clauses for $y_{|C|} = 1$, being $y_{|C|}$ the output of the circuit.

In this construction, if $f(\underline{x}) = 1$ then there exists a unique $\underline{y}$ such that $\phi(\underline{x}, \underline{y}) = 1$, whereas if $f(\underline{x}) = 0$ then $\phi(\underline{x}, \underline{y}) = 0$ for all $\underline{y}$.

By using a minor variant of the reduction from 3-Sat to SubsetSum (see, for example, [6] and [4, pag. 223]), we can reduce $\phi(\underline{x}, \underline{y})$ to an instance $I_n(\underline{\alpha}, \underline{\beta}, \underline{\gamma}; t)$ of SubsetSum, with $|\underline{\gamma}| = n^{O(1)}$, such that

– $\phi(\underline{x}, \underline{y}) = 1$ implies $\exists! \underline{w}$ such that $\underline{x} \cdot \underline{\alpha} + \underline{y} \cdot \underline{\beta} + \underline{w} \cdot \underline{\gamma} = t$;
– $\phi(\underline{x}, \underline{y}) = 0$ implies $\underline{x} \cdot \underline{\alpha} + \underline{y} \cdot \underline{\beta} + \underline{w} \cdot \underline{\gamma} \neq t$ for every $\underline{w}$.

Let now $\xi(I_n)$ and $\xi'(I_n)$ be the Lohrey strings associated with the instance $I_n(\underline{\alpha}, \underline{\beta}, \underline{\gamma}; t)$. Then, we have two words of length $2^{n+m+|\gamma|}$ such that

– $f(\underline{x}) = 1 \implies \exists! z \in \{0,1\}^{m+|\gamma|} \mid \xi(I_n)[b(xz)] = \xi'(I_n)[b(xz)] = 1$;
– $f(\underline{x}) = 0 \implies \forall z \in \{0,1\}^{m+|\gamma|} \mid \xi(I_n)[b(xz)] \wedge \xi'(I_n)[b(xz)] = 0$;

By Lemma 2, $g(\xi(I_n)), g(\xi'(I_n)) = n^{O(1)}$. □

Now, we are able to prove the main result of this section:

Theorem 2. *For all $n > 0$, there exist two words w_n and w'_n of equal length such that $g(w_n), g(w'_n) = n^{O(1)}$ and $g(w_n \wedge w'_n) = \Omega(2^{n/2})$.*

Proof. Let C be the circuit computing the boolean function

$$f(\underline{x}, \underline{y}) = \begin{cases} 1 & \text{if } b(x) = b(y)^2 \\ 0 & \text{if } b(x) \neq b(y)^2 \end{cases}$$

A circuit C for f can be realized with $O(|x|^2)$ variables and $O(|x|^2)$ 3-clauses by iterated sums.

Consider the instance $I_n(\underline{\alpha}, \underline{\beta}, \underline{\gamma}; t)$ of SubsetSum defined for C as in Lemma 3, fix the representation $q(s)$ of the sth perfect square s^2 and let $z(s)$ be the unique string such that $\underline{q(s)} \cdot \underline{\alpha} + \underline{z(s)} \cdot (\underline{\beta} \odot \underline{\gamma}) = t$. Let now $\xi(n)$ and $\xi'(n)$ be the Lohrey strings associated with I_n and let be $\xi = \xi(n) \wedge \xi'(n)$. The position t_s of the sth 1 in ξ is equal to $b(q(s)z(s))$. Since $z(s)$ is unique, we have

$$s^2 2^M \leq t_s < s^2 2^M + 2^M,$$

where $M = |z(s)|$. It follows that

$$s2^{M+1} \ \leq \ t_{s+1} - t_s \ < \ (s+1)2^{M+1}.$$

This implies that $t_{s+1} - t_s \neq t_{j+1} - t_j$ whenever $s \neq j$. As a consequence, fixing $\hat{s} = \max\{s \mid s^2 < 2^n\}$, it holds

$$|\text{Fact}\,(\xi) \cap 10^*1| \ = \ \left|\{10^{t_{s+1}-t_s}1 \mid 1 \leq s \leq \hat{s}\}\right| \ = \ \hat{s} \ \geq \ 2^{n/2} - 1.$$

By Lemma 1, we have $|LZ(\xi)| \geq |\text{Fact}\,(\xi) \cap 10^*1|$. Hence $g(\xi) = \Omega(2^{n/2})$, while $g(\xi(n)), g(\xi'(n)) = n^{O(1)}$ by Lemma 2. □

In Example 1, we have a rational transducer A such that $A(x \sqcup\!\sqcup y) = x \wedge y$. Hence, by exploiting Theorem 1, the previous result can be extended to the literal shuffle of words.

Corollary 2. *For all $n > 0$, there exist two words w_n and w'_n of equal length such that $g(w_n), g(w'_n) = n^{O(1)}$ and $g(w_n \sqcup\!\sqcup w'_n) = \Omega(2^{n/2})$.*

5.1 Picture Straight-Line Programs

A natural representation of 2D texts can be obtained by using a 2D extension of SLPs. Informally, a binary picture of width M and height N is a matrix $T \in \{0,1\}^{N \times M}$. We refer to the rows and columns of a picture as its sections. A 2D-SLP of size n is a sequence of labelled instructions of the form

$$X_k \ = \ 1 \mid 0 \mid X_i \oslash X_j \mid X_i \ominus X_j, \quad 0 < i, j < k \quad k = 1, \ldots, n.$$

The operator $\oslash$ is the horizontal concatenation between two pictures of equal height, while $\ominus$ is the vertical concatenation of two pictures of equal width. The output of a 2D-SLP Φ is a binary picture $T = \text{eval}(\Phi)$, obtained performing all the concatenations in Φ; the descriptional complexity $g(T)$ of a picture T is the size of the smallest 2D-SLP generating T.

The structure of 2D-SLPs is more complex than that of SLPs. In particular, while the factors of logarithmically compressible words are still logarithmically compressible, an analogous property does not hold for subpictures of pictures. For instance, in [2] it is proved that, for each n, there exists a picture T_n with $g(T_n) = n$ having at least one section w_n with $g(w_n) = 2^{\Omega(n)}$.

Here, we prove a stronger result on the sections of compressed pictures.

Theorem 3. *Let T be a $N \times M$ picture and let c_i be its i-th column and r_j its j-th row. Then, $g(c_i) \leq g(T) \times \min\{i, M-i\}$ and $g(r_j) \leq g(T) \times \min\{j, N-j\}$.*

Proof. Let $\Phi = (X_1, \ldots, X_n)$ be a 2D-SLP such that $\text{eval}(\Phi) = T$. Then, we construct a SLP Φ_s for the sth column of T in the following way. Without loss of generality, suppose that $s < \lfloor M/2 \rfloor$ and, for each variable X_k of Φ, define s variables of the form

$$X_k(s) \;=\; \begin{cases} \sigma & \text{if } \; X_k = \sigma; \\ X_i(s)X_j(s) & \text{if } \; X_k = X_i \ominus X_j; \\ X_i(s) & \text{if } \; X_k = X_i \oslash X_j \text{ and } s \le |X_i|\,; \\ X_j(s - |X_i|) & \text{if } \; X_k = X_i \oslash X_j \text{ and } s > |X_i|\,. \end{cases}$$

Then, $|\Phi_s| = s \times |\Phi|$ and $\mathrm{eval}_{\Phi_s}(X_n(s))$ is the sth column of T. The technique can be easily adapted for the columns of position grater that $\lceil M/2 \rceil$ and for all the rows of T. □

The previous result gives an upper bound for the compression rate of the sections of a picture. However, we would ask how much strict is this bound. Next theorem proves that the bound is, in some sense, optimal.

Theorem 4. *There exists an infinite number of pictures $\{T_n\}_{n \in \mathbf{N}}$ such that, for each column c_i of T_n in position i, it holds $g(c_i) = (g(T_n) \times \min\{i, M - i\})^{\Omega(1)}$.*

Proof. Consider the Lohrey strings $\xi(n), \xi'(n)$ related with the instance of SubsetSum in the proof of Th. 2 and let $l = |\xi(n)| = |\xi'(n)|$. By Lemma 2, we have two polynomial size SLPs Φ and Φ' such that $\mathrm{eval}(\Phi) = \xi(n)$ and $\mathrm{eval}(\Phi') = \xi'(n)$. Moreover, let Z_i be the $2^i \times 2^{i-1}$ picture containing all zeros. For $i = 1, \dots, k$, all the Z_i can be compressed by a SLP of size $O(k)$. Then, we can construct a polynomial size SLP for the picture T described recursively by

$$\begin{aligned} X_1 &= \xi(n) \ominus \xi'(n); \\ X_{i+1} &= (Z_i \oslash X_i) \ominus (X_i \oslash Z_i), \quad i = 1, \dots, 2n. \end{aligned}$$

In this way, we obtain a picture T of size $2^{2n+1} \times (l + 2^{2n} - 1)$ of the form

$$\begin{bmatrix} 0 & \cdots & \cdots & \cdots & \cdots & 0 & a_1 & a_2 & \cdots & a_l \\ \vdots & \cdots & \cdots & \cdots & \cdots & 0 & b_1 & b_2 & \cdots & b_l \\ \vdots & \cdots & \cdots & \cdots & 0 & a_1 & a_2 & \cdots & a_l & 0 \\ \vdots & \cdots & \cdots & \cdots & 0 & b_1 & b_2 & \cdots & b_l & 0 \\ \vdots & \cdots & \cdots & 0 & a_1 & a_2 & \cdots & a_l & 0 & \vdots \\ \vdots & \cdots & \cdots & 0 & b_1 & b_2 & \cdots & b_l & 0 & \vdots \\ 0 & \cdots & \iddots & \iddots & \iddots & \iddots & \iddots & \iddots & \vdots & \vdots \\ a_1 & a_2 & \cdots & a_l & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ b_1 & b_2 & \cdots & b_l & 0 & \cdots & \cdots & \cdots & \cdots & 0 \end{bmatrix}$$

with $\xi(n) = a_1 \cdots a_l$ and $\xi'(n) = b_1 \cdots b_l$. Clearly, $g(T) = n^{\Theta(1)}$. Note that every column c_i of T in position i such that $l \le i < 2^{2n}$ contains all zeros except a factor $\xi(n) \sqcup\!\sqcup \xi'(n)$. So, $g(c_i) = n^{\Theta(1)}\, g(\xi(n) \sqcup\!\sqcup \xi'(n)) = (n2^n)^{\Theta(1)}$.

The cases $1 \le i < l$ and $2^{2n} \le i < 2^{2n} + l$ are symmetric, so we focus on the former. In this case, only the prefix $p_i = (\xi(n) \sqcup\!\sqcup \xi'(n))[1, 2i]$ appears at the end of c_i, while the rest of the column is all zeros. By Corollary 1, $O(g(p_i)) = g(\xi[1, 2i])$, where $\xi = \xi(n) \wedge \xi'(n)$. The structure of ξ is such that

$$g(\xi[1, 2i]) \;\ge\; |\mathrm{Fact}\,(\xi[1, 2i]) \cap 10^*1| \;=\; \Omega(\sqrt{i}).$$

Hence, again $g(c_i) = (ni)^{\Omega(1)}$. $\square$

Obviously, the same result holds for the case of rows of pictures.

6 Pattern Matching and Literal Shuffle

In this section, we represent a word $w = \text{eval}(\Phi) \sqcup\!\sqcup \text{eval}(\Psi)$ by means of the pair (Φ, Ψ) with size $|\Phi| + |\Psi|$ and study the compressed pattern matching problem in this representation. More formally, the problem can be stated as

PROBLEM:	*Compressed Pattern Matching with* $\sqcup\!\sqcup$ *(CPM$_{\sqcup\!\sqcup}$)*
INSTANCE:	Four SLPs Φ, Ψ, Φ', Ψ', such that $|\text{eval}(\Phi)| = |\text{eval}(\Phi')|$ and $|\text{eval}(\Psi)| = |\text{eval}(\Psi')|$;
QUESTION:	is $\text{eval}(\Psi) \sqcup\!\sqcup \text{eval}(\Psi')$ a factor of $\text{eval}(\Phi) \sqcup\!\sqcup \text{eval}(\Phi')$?

It can be easily observed that CPM$_{\sqcup\!\sqcup}$ is reducible to the Compressed Pattern Matching for pictures composed by two lines. So, it appears more difficult than Compressed Pattern Matching for words, solvable in polynomial time ([9]), but easier than Compressed Pattern Matching for pictures, which is Σ_2^P-complete ([2]).

CPM$_{\sqcup\!\sqcup}$ is clearly in NP. Moreover, it is NP-complete even if the pattern is the string 11.

Theorem 5. *The problem of deciding, given two SLPs Φ and Φ', whether* 11 *is a factor of eval$(\Phi) \sqcup\!\sqcup$ eval(Φ') is NP-hard.*

Proof. By reduction to SUBSETSUM. Let $\xi(I)$ and $\xi'(I)$ be the Lohrey strings associated with an instance I of SUBSETSUM and let Φ and Φ' be the associated SLPs of smallest size. Then, define a deterministic rational transducer A such that $A(x) = 0x[1]0x[2] \cdots 0x[|x|]$, for every word x. By Th. 1, $g(A(x)) = O(g(x))$. Moreover, $A(\text{eval}(\Phi)) \sqcup\!\sqcup A(\text{eval}(\Phi'))$ contains the factor 11 if and only if the instance I of SUBSETSUM admits a solution. $\square$

Despite this hardness result, we exhibit an algorithm for CPM$_{\sqcup\!\sqcup}$ working in polynomial time if the length of the pattern is polynomially related with that of the text.

We recall some notation about SLPs used in [9]. The *positions* 0 and n in a word $w \in \Sigma^n$ are the points immediately before $w[1]$ and after $w[n]$, respectively, while a position i, with $1 \leq i < n$ is the point between $w[i]$ and $w[i+1]$. A factor $w[i,j]$ touches a position k in w if $i - 1 \leq k \leq j$. Given a nonterminal symbol X_k in a SLP Φ such that $X_k = X_i X_j$, its *cut position* is $|\text{eval}_\Phi(X_i)|$.

Finally, by the triple of nonnegative integers (p, d, r) we codify the *arithmetical progression* $\{p, p+d, p+2d, \ldots, p+rd\}$ and recall the following result.

Lemma 4. *Let (p, d, r) and (p', d', r') be two arithmetical progressions, where p, d, r, p', d', r' are n-bits integers. Then, deciding if their intersection is empty requires $O(n^2)$ time.*

Proof. The problem of deciding if $(p,d,r) \cap (p',d',r')$ is the empty set consists of verifying the existence of two integers x, y such that

1. $p + dx = p' + d'y$;
2. $0 \leq x \leq r$ and $0 \leq y \leq r'$.

Such equations are equivalent to the diophantine equation $Ax - By = C$, where $c = MCD(d, d', p' - p)$ and $A = d/c$, $B = d'/c$, $C = (p' - p)/c$.

If $MCD(A,B) > 1$, then the previous equation has no solution and $(p,d,r) \cap (p',d',r') = \emptyset$. Otherwise, one solution (x_0, y_0) can be obtained by computing the $(h-1)$th convergent, where h is the number of terms in the continued fraction for A/B ([14]).

All the other solutions are of the form $x = c(x_0 + kB)$, $y = c(y_0 + kA)$. By setting

$$\begin{aligned}\underline{k} &= \min\{k \mid 0 \leq c(x_0 + kB)\},\\ \overline{k} &= \max\{k \mid c(x_0 + kB) \leq r\},\\ \underline{k}' &= \min\{k \mid 0 \leq c(y_0 + kA)\},\\ \overline{k}' &= \max\{k \mid c(y_0 + kA) \leq r\},\end{aligned}$$

one can conclude that $(p,d,r) \cap (p',d',r') \neq \emptyset$ if and only if $[\underline{k}, \overline{k}] \cap [\underline{k}', \overline{k}'] \neq \emptyset$.

The most expensive task in this process is the computation of the convergent of a fraction of two n-bits integers, which takes $O(n^2)$ time. □

Many compressed pattern matching algorithms are based on the following ([5])

Lemma 5. *Given two words w and v, all the occurrences of v in w touching a fixed position form a single arithmetical progression.*

Some compressed pattern matching algorithms use as a data structure the so called AP-table, which we recall in a simplified version. Given two SLPs Φ and Ψ, the AP-table for Φ and Ψ is a vector where, for every symbol X_k in Φ, the k-th entry is the (possibly empty) arithmetical progression $(p[X_k], d[X_k], r[X_k])$ identifying the starting positions of the occurrences of eval(Ψ) that touch the cut position of X_k. The AP-table for Φ and Ψ is computable in time $O(|\Phi|^3 \times |\Psi|)$ ([9]).

Given a SLP Φ having variables $X_1, \ldots, X_n$, consider the following partial function $t : \{X_1, \ldots, X_n\} \longrightarrow \{X_1, \ldots, X_n\}$, such that

$$t(X_k) = \begin{cases} X_i & \text{if } X_k = X_iX_j \text{ and } |\text{eval}_\Phi(X_i)| \geq |\text{eval}_\Phi(X_k)|/2,\\ X_j & \text{if } X_k = X_iX_j \text{ and } |\text{eval}_\Phi(X_i)| < |\text{eval}_\Phi(X_k)|/2,\\ \perp & \text{otherwise.}\end{cases}$$

Now, select the path $z_1, \ldots, z_j$ in the derivation tree of eval(Φ), defined as $z_1 = X_n$, $z_{i+1} = t(z_i)$ for $1 \leq i < j$, where z_j is the first occurrence of $\perp$. Then, this path identifies a sequence of factors $f_1, \ldots, f_j$ of eval(Φ), each of which is a prefix or a suffix of its predecessor. Hence, their starting positions $g_1, \dots, g_j$ can be computed in time $O(|\Phi|^2)$.

Now, suppose that $|\text{eval}(\Psi)| > |\text{eval}(\Phi)|/2$. Then, the possible starting positions of $\text{eval}(\Psi)$ in $\text{eval}(\Phi)$ are the elements defined in the arithmetical progressions

$$ar_i(\Phi, \Psi) = (g_i + p[z_i], d[z_i], r[z_i]), \quad \text{for } i = 1, \dots, j.$$

Theorem 6. *Let Φ, Ψ, Φ' and Ψ' be four SLPs such that $|eval(\Phi)| = |eval(\Phi')| = N$ and $|eval(\Psi)| = |eval(\Psi')| = M$. Then, the CPM$_\sqcup$ problem can be solved in time $O(Nn^4/M)$, where $n = |\Phi| + |\Psi| + |\Phi'| + |\Psi'|$.*

Proof. Suppose that $N < 2M$. Then,

$$\text{eval}(\Psi) \sqcup \text{eval}(\Psi') \in \text{Fact}\,(\text{eval}(\Phi) \sqcup \text{eval}(\Phi'))$$

if and only if, for some integers i, s, at least one of the following facts hold:

- $ar_i(\Phi, \Psi) \cap ar_s(\Phi', \Psi') \neq \emptyset$;
- $ar_i(\Phi', \Psi) \cap ar'_s(\Phi, \Psi') \neq \emptyset$;

where $ar'_s(\Phi, \Psi')$ is $ar_s(\Phi, \Psi')$ left-shifted by one, i.e., if $ar_s(\Phi, \Psi') = (g+p, d, r)$, then $ar'_s(\Phi, \Psi') = (g+p-1, d, r)$. By Lemma 4, non-emptiness of each intersection can be verified in $O(n^2)$ time and, since i and s range over $[1, n]$, we can solve the problem in time $O(n^4)$.

If, on the contrary, $N \geq 2M$, then, for $0 \leq i \leq 2N/M - 2$, we construct the SLPs Φ_i, such that

$$\text{eval}(\Phi_i) = \text{eval}(\Phi)[i\lceil M/2\rceil + 1, \min\{N, (i+3)\lceil M/2\rceil\}]$$

and we do the same for Φ'. This construction requires $O(n^2 N/M)$ time and guarantees that the following sentences are equivalent:

1. The word $\text{eval}(\Psi) \sqcup \text{eval}(\Psi')$ is a factor of $\text{eval}(\Phi) \sqcup \text{eval}(\Phi')$.
2. There exists k $(0 \leq k \leq N - M)$ such that at least one of these facts hold:
   - $\text{eval}(\Psi)$ is a factor of $\text{eval}(\Phi)$ and $\text{eval}(\Psi')$ is a factor of $\text{eval}(\Phi')$, both starting in position k;
   - $\text{eval}(\Psi)$ is the factor of $\text{eval}(\Phi')$ starting in position k and $\text{eval}(\Psi')$ is the factor of $\text{eval}(\Phi)$ starting in position $k+1$.
3. There exists i $(0 \leq i \leq 2N/M - 2)$ such that $\text{eval}(\Psi) \sqcup \text{eval}(\Psi')$ is a factor of $\text{eval}(\Phi_i) \sqcup \text{eval}(\Phi'_i)$.

Since $|\Phi_i| = |\Phi'_i| < 2M$, for every i, we can verify whether $\text{eval}(\Psi) \sqcup \text{eval}(\Psi')$ is a factor of $\text{eval}(\Phi_i) \sqcup \text{eval}(\Phi'_i)$ in time $O(n^4)$. Hence, the problem is solvable in time $O(n^4 N/M)$. $\square$

7 Conclusions

We investigated the possibility of performing rational transformations and the literal shuffle of words compressed via SLPs, without full unpacking. We proved that the last operation does not preserve the compression rate; hence, some techniques like Cooley-Tukey algorithm for FFT can not be applied in a compressed context. On the other hand, rational transformations can be performed without fully uncompressing the SLPs in input.

These results lead to a deeper insight into the relations between SLPs for words and SLPs for pictures. Indeed we showed that the descriptional complexity of the sections of a picture can strongly depend on their distance from the borders.

The literal shuffle has been finally exploited as a compressed representation of pictures having two lines. The associated compressed pattern matching problem lies in the half way between the same problems for compressed words and for compressed pictures. We proposed a parameter-tractable algorithm working in polynomial time, where the parameter is the ratio between the length of the text and that of the pattern.

Acknowledgements We would like to thank Antonio Restivo for some useful discussions.

References

1. Béatrice Bérard. Literal shuffle. *Theoret. Comput. Sci.*, 51(3):281–299, 1987.
2. Piotr Berman, Marek Karpinski, Lawrence L. Larmore, Wojciech Plandowski, and Wojciech Rytter. On the complexity of pattern matching for highly compressed two-dimensional texts. In *Combinatorial pattern matching (Aarhus, 1997)*, volume 1264 of *Lecture Notes in Comput. Sci.*, pages 40–51. Springer, Berlin, 1997.
3. James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Math. Comp.*, 19:297–301, 1965.
4. Michael R. Garey and David S. Johnson. *Computers and intractability.* W. H. Freeman and Co., San Francisco, Calif., 1979. A guide to the theory of NP-completeness, A Series of Books in the Mathematical Sciences.
5. Leszek Gąsieniec, Marek Karpinski, Wojciech Plandowski, and Wojciech Rytter. Randomized efficient algorithms for compressed strings: the finger-print approach (extended abstract). In *Combinatorial pattern matching (Laguna Beach, CA, 1996)*, volume 1075 of *Lecture Notes in Comput. Sci.*, pages 39–49. Springer, Berlin, 1996.
6. Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations (Proc. Sympos., IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y., 1972)*, pages 85–103. Plenum, New York, 1972.
7. Abraham Lempel and Jacob Ziv. On the complexity of finite sequences. *IEEE Trans. Information Theory*, IT-22(1):75–81, 1976.
8. Abraham Lempel and Jacob Ziv. A universal algorithm for sequential data compression. *IEEE Trans. Information Theory*, IT-23(3):337–343, 1977.
9. Yury Lifshits. Processing compressed texts: A tractability border. In *Combinatorial Pattern Matching*, volume 4580 of *Lecture Notes in Comput. Sci.*, pages 228–240. Springer, Berlin, 2007.

10. Markus Lohrey. Word problems on compressed words. In *Automata, languages and programming*, volume 3142 of *Lecture Notes in Comput. Sci.*, pages 906–918. Springer, Berlin, 2004.
11. Markus Lohrey. Word problems and membership problems on compressed words. *SIAM J. Comput.*, 35(5):1210–1240 (electronic), 2006.
12. N. Markey and Ph. Schnoebelen. A PTIME-complete matching problem for SLP-compressed words. *Inform. Process. Lett.*, 90(1):3–6, 2004.
13. Masamichi Miyazaki, Ayumi Shinohara, and Masayuki Takeda. An improved pattern matching algorithm for strings in terms of straight-line programs. *J. Discrete Algorithms (Oxf.)*, 1(1):187–204, 2000.
14. C. D. Olds. *Continued fractions.* Random House, New York, 1963.
15. Wojciech Plandowski. Testing equivalence of morphisms on context-free languages. In *Algorithms—ESA '94 (Utrecht)*, volume 855 of *Lecture Notes in Comput. Sci.*, pages 460–470. Springer, Berlin, 1994.
16. Wojciech Plandowski and Wojciech Rytter. Complexity of language recognition problems for compressed words. In *Jewels are forever*, pages 262–272. Springer, Berlin, 1999.
17. Wojciech Rytter. Application of Lempel-Ziv factorization to the approximation of grammar-based compression. *Theoret. Comput. Sci.*, 302(1-3):211–222, 2003.