Approximation ratio of RePair*

Danny Hucke\textsuperscript{1}, Artur Jeż\textsuperscript{2}, and Markus Lohrey\textsuperscript{1}

\textsuperscript{1}University of Siegen, Germany
\textsuperscript{2}University of Wroclaw, Poland

Abstract

In a seminal paper of Charikar et al. on the smallest grammar problem, the authors derive upper and lower bounds on the approximation ratios for several grammar-based compressors. Here we improve the lower bound for the famous RePair algorithm from $\Omega(\sqrt{\log n})$ to $\Omega(\log n / \log \log n)$. The family of words used in our proof is defined over a binary alphabet, while the lower bound from Charikar et al. needs an alphabet of logarithmic size in the length of the provided words.

1 Introduction

The idea of grammar-based compression is based on the fact that in many cases a word $w$ can be succinctly represented by a context-free grammar that produces exactly $w$. Such a grammar is called a straight-line program (SLP) for $w$. In the best case, one gets an SLP of size $O(\log n)$ for a word of length $n$, where the size of an SLP is the total length of all right-hand sides of the rules of the grammar. A grammar-based compressor is an algorithm that produces for a given word $w$ an SLP $A$ for $w$, where, of course, $A$ should be smaller than $w$. Grammar-based compressors can be found at many places in the literature. Probably the best known example is the classical LZ78-compressor of Lempel and Ziv \cite{LZ78}. Indeed, it is straightforward to transform the LZ78-representation of a word $w$ into an SLP for $w$. Other well-known grammar-based compressors are BISECTION \cite{Bis}, SEQUITUR \cite{Sequ}, and RePair \cite{Rep}, just to mention a few.

One of the first appearances of straight-line programs in the literature are \cite{Bers,Brel}, where they are called word chains (since they generalize addition chains from numbers to words). In \cite{Bers,Brel}, Berstel and Brlek prove that the function $g(k,n) = \max\{g(w) \mid w \in \{1, \ldots, k\}^n\}$, where $g(w)$ is the size of a smallest SLP for the word $w$, is in $\Theta(n/\log_k n)$. Note that $g(k,n)$ measures the worst case SLP-compression over all words of length $n$ over a $k$-letter alphabet. The first systematic investigations of grammar-based compressors are \cite{Char1,Char2}. Whereas in \cite{Char1}, grammar-based compressors are used for universal lossless compression (in the information-theoretic sense), Charikar et al. study in \cite{Char1} the worst case approximation ratio of grammar-based compressors. For a given grammar-based compressor $C$ that computes from a given word $w$ an SLP $C(w)$ for $w$ one defines the approximation ratio of $C$ on $w$ as the quotient of the size of $C(w)$ and

\*The second and third author were supported by the DFG research grant LO 748/10-1.
the size \( q(w) \) of a smallest SLP for \( w \). The approximation ratio \( \alpha_C(n) \) is the maximal approximation ratio of \( C \) among all words of length \( n \) over any alphabet. In [3] the authors compute upper and lower bounds for the approximation ratios of several grammar-based compressors (among them are the compressors mentioned above). The contribution of this paper is the improvement of the lower bound for RePair from \( \Omega(\sqrt{\log n}) \) to \( \Omega(\log n / \log \log n) \). While in [3] the lower bound needs an unbounded alphabet (the alphabet grows logarithmically in the length of the presented words) our family of words is defined over a binary alphabet.

RePair works by repeatedly searching for a digram \( d \) (a string of length two) with the maximal number of non-overlapping occurrences in the current text and replacing all these occurrences by a new nonterminal \( A \). Moreover, the rule \( A \to d \) is added to the grammar. RePair is one of the so-called global grammar-based compressor from [3] for which the approximation ratio seems to be very hard to analyze. Charikar et al. prove for all global grammar-based compressors an upper bound of \( \mathcal{O}((n/\log n)^{2/3}) \) for the approximation ratio. Note that the gap to our improved lower bound \( \Omega(\log n / \log \log n) \) is still large.

Related work. The theoretically best known grammar-based compressors with a polynomial (in fact, linear) running time achieve an approximation ratio of \( \mathcal{O}(\log n) \) [3, 9, 10, 17]. In [8], the precise (up to constant factors) approximation ration for BISECTION (resp., LZ78) was shown to be \( \Theta((n/\log n)^{1/2}) \) (resp., \( \Theta((n/\log n)^{2/3}) \)). In [15] the authors prove that RePair combined with a simple binary encoding of the grammar compresses every word \( w \) over an alphabet of size \( \sigma \) to at most \( 2H_k(w) + o(|w| \log \sigma) \) bits, for any \( k = o(\log_{\sigma}|w|) \), where \( H_k(w) \) is the \( k \)-th order entropy of \( w \).

There is also a bunch of papers with practical applications for RePair: web graph compression [4], bit maps [14], compressed suffix trees [7]. Some practical improvements of RePair can be found in [6].

2 Preliminaries

Let \( [1,k] = \{1, \ldots , k\} \). Let \( w = a_1 \cdots a_n \) \((a_1, \ldots , a_n \in \Sigma)\) be a word or string over a finite alphabet \( \Sigma \). The length \( |w| \) of \( w \) is \( n \) and we denote by \( \varepsilon \) the word of length 0. We define \( w[i] = a_i \) for \( 1 \leq i \leq |w| \) and \( w[i:j] = a_i \cdots a_j \) for \( 1 \leq i \leq j \leq |w| \). Let \( \Sigma^* = \Sigma^* \setminus \{\varepsilon\} \) be the set of nonempty words. For \( w \in \Sigma^* \), we call \( v \in \Sigma^* \) a factor of \( w \) if there exist \( x,y \in \Sigma^* \) such that \( w = xy \). If \( x = \varepsilon \), then we call \( v \) a prefix of \( w \). For words \( w_1, \ldots , w_n \in \Sigma^* \), we further denote by \( \prod_{i=1}^{n} w_i \) the word \( w_jw_{j+1} \cdots w_n \) if \( j \leq n \) and \( \varepsilon \) otherwise.

A straight-line program, briefly SLP, is a context-free grammar that produces a single word \( w \in \Sigma^* \). Formally, it is a tuple \( \mathcal{A} = (N, \Sigma, P, S) \), where \( N \) is a finite set of nonterminals with \( N \cap \Sigma = \emptyset \), \( S \in N \) is the start nonterminal, and \( P \) is a finite set of productions (or rules) of the form \( A \to w \) for \( A \in N \), \( w \in (N \cup \Sigma)^* \) such that:

- For every \( A \in N \), there exists exactly one production of the form \( A \to w \), and
- the binary relation \( \{(A,B) \in N \times N \mid (A \to w) \in P, B \text{ occurs in } w\} \) is acyclic.
Every nonterminal \( A \in N \) produces a unique string \( \text{val}_A(A) \in \Sigma^+ \). The string defined by \( A \) is \( \text{val}(A) = \text{val}_A(S) \). We omit the subscript \( A \) when it is clear from the context. The size of the SLP \( A \) is \( |A| = \sum_{(A \rightarrow w) \in P} |w| \). We denote by \( g(w) \) the size of a smallest SLP producing the word \( w \in \Sigma^+ \). We will use the following lemma:

**Lemma 1** (\[3, Lemma 3\]). A string \( w \) contains at most \( g(w) \cdot k \) distinct factors of length \( k \).

A grammar-based compressor \( C \) is an algorithm that computes for a nonempty word \( w \) an SLP \( C(w) \) such that \( \text{val}(C(w)) = w \). The **approximation ratio** \( \alpha_C(w) \) of \( C \) for an input \( w \) is defined as \( |C(w)|/g(w) \). The worst-case approximation ratio \( \alpha_C(k, n) \) of \( C \) is the maximal approximation ratio over all words of length \( n \) over an alphabet of size \( k \):

\[
\alpha_C(k, n) = \max_{w \in [1, k]^n} \{ |C(w)|/g(w) \}
\]

If the alphabet size is unbounded, i.e., if we allow alphabets of size \(|w|\), then we write \( \alpha_C(n) \) instead of \( \alpha_C(n, n) \).

### 3 RePair

For a given SLP \( A = (N, \Sigma, P, S) \), a word \( \gamma \in (N \cup \Sigma)^+ \) is called a maximal string of \( A \) if

- \( |\gamma| \geq 2 \),
- \( \gamma \) appears at least twice without overlap in the right-hand sides of \( A \),
- and no strictly longer word appears at least as many times on the ride-hand sides of \( A \) without overlap.

A global grammar-based compressor starts on input \( w \) with the SLP \( A = (\{S\}, \Sigma, \{S \rightarrow w\}, S) \). In each round, the algorithm selects a maximal string \( \gamma \) of \( A \) and updates \( A \) by replacing a largest set of a pairwise non-overlapping occurrences of \( \gamma \) in \( A \) by a fresh nonterminal \( X \). Additionally, the algorithm introduces the rule \( X \rightarrow \gamma \). The algorithm stops when no maximal string occurs.

The global grammar-based compressor RePair \[13\] selects in each round a most frequent maximal string. Note that the replacement is not unique, e.g. the word \( a^5 \) with the maximal string \( \gamma = aa \) yields SLPs with rules \( S \rightarrow XXa, X \rightarrow aa \) or \( S \rightarrow XaX, X \rightarrow aa \) or \( S \rightarrow aXX, X \rightarrow aa \). We assume the first variant in this paper, i.e. maximal strings are replaced from left to right.

The above description of RePair is taken from \[3\]. In most papers on RePair the algorithm works slightly different: It replaces in each step a digram (a string of length two) with the maximal number of pairwise non-overlapping occurrences in the right-hand sides. For example, for the string \( w = abcabc \) this produces the SLP \( S \rightarrow BB, B \rightarrow Ac, A \rightarrow ab \), whereas the RePair-variant from \[3\] produces the smaller SLP \( S \rightarrow AA, A \rightarrow abc \).

The following lower and upper bounds on the approximation ratio of RePair were shown in \[3\]:

- \( \alpha_{\text{RePair}}(n) \in \Omega(\sqrt{\log n}) \)
• \( o_{\text{RePair}}(2, n) \in \mathcal{O}\left((n/ \log n)^{2/3}\right) \)

The proof of the lower bound in \([3]\) assumes an alphabet of unbounded size. To be more accurate, the authors construct for every \( k \) a word \( w_k \) of length \( \Theta(\sqrt{2^k}) \) over and alphabet of size \( \Theta(k) \) such that \( g(w) \in O(k) \) and \( \text{RePair} \) produces a grammar of size \( \Omega(k^{3/2}) \) for \( w_k \). We will improve this lower bound using only a binary alphabet. To do so, we first need to know how \( \text{RePair} \) compresses unary words.

**Example 1** (unary inputs). \( \text{RePair} \) produces on input \( a^{27} \) the SLP with rules \( X_1 \rightarrow aa, X_2 \rightarrow X_1X_1, X_3 \rightarrow X_2X_2 \) and \( S \rightarrow X_3X_3X_3X_1a \), where \( S \) is the start nonterminal. For the input \( a^{22} \) only the start rule \( S \rightarrow X_3X_3X_2X_1 \) is different.

In general, \( \text{RePair} \) creates on unary input \( a^m \) (\( m \geq 4 \)) the rules \( X_1 \rightarrow aa, X_i \rightarrow X_{i-1}X_{i-1} \) for \( 2 \leq i \leq \lceil \log m \rceil - 1 \) and a start rule, which is strongly related to the binary representation of \( m \) since each nonterminal \( X_i \) produces the word \( a^i \). To be more accurate, let \( b_{\lceil \log m \rceil}b_{\lceil \log m \rceil-1} \cdots b_1b_0 \) be the binary representation of \( m \) and define the mappings \( f_i (i \geq 0) \) by:

- \( f_0 : \{0, 1\} \rightarrow \{a, \varepsilon\} \) with \( f_0(1) = a \) and \( f_0(0) = \varepsilon \),
- \( f_i : \{0, 1\} \rightarrow \{X_i, \varepsilon\} \) with \( f_i(1) = X_i \) and \( f_i(0) = \varepsilon \) for \( i \geq 1 \).

Then the start rule produced by \( \text{RePair} \) on input \( a^m \) is

\[
S \rightarrow X_{\lceil \log m \rceil-1}X_{\lceil \log m \rceil-1}f_{\lceil \log m \rceil-1}(b_{\lceil \log m \rceil-1}) \cdots f_1(b_1)f_0(b_0).
\]

This means that the symbol \( a \) only occurs in the start rule if \( b_0 = 1 \), and the nonterminal \( X_i \) (\( 1 \leq i \leq \lceil \log m \rceil - 2 \)) occurs in the start rule if and only if \( b_i = 1 \). Since \( \text{RePair} \) only replaces words with at least two occurrences, the most significant bit \( b_{\lceil \log m \rceil} = 1 \) is represented by \( X_{\lceil \log m \rceil-1}X_{\lceil \log m \rceil-1} \). Note that for \( 1 \leq m \leq 3 \), \( \text{RePair} \) produces the trivial SLP \( S \rightarrow a^m \).

### 4 Main result

The main result of this paper states:

**Theorem 1.** \( \alpha_{\text{RePair}}(2, n) \in \Omega(\log n/ \log \log n) \)

**Proof.** We start with a binary De-Brujin sequence \( B_{\lceil \log k \rceil} \in \{0, 1\}^* \) of length \( 2^{\lceil \log k \rceil} \) such that each factor of length \( \lceil \log k \rceil \) occurs at most once \([2]\). We have \( k \leq |B_{\lceil \log k \rceil}| < 2k \). Note that De-Brujin sequences are not unique, so without loss of generality let us fix a De-Brujin sequence which starts with 1 for the remaining proof. We define a homomorphism \( h : \{0, 1\}^* \rightarrow \{0, 1\}^* \) by \( h(0) = 01 \) and \( h(1) = 10 \). The words \( w_k \) of length \( 2k \) are defined as

\[
w_k = h(B_{\lceil \log k \rceil}[1 : k]).
\]

For example for \( k = 4 \) we can take \( B_2 = 1100 \), which yields \( w_4 = 10100101 \). We will analyze the approximation ratio of \( \text{RePair} \) for the binary words

\[
s_k = \prod_{i=1}^{k-1} \left(a^{w_k[1:k+i]}b\right)^{a^{w_k}} = a^{w_k[1:k+1]}b_a^{w_k[1:k+2]}b_a^{w_k[1:2k-1]}ba^{w_k},
\]

4
where the prefixes $w_k[1 : k + i]$ for $1 \leq i \leq k$ are interpreted as binary numbers. For example we have $s_4 = a^{20}ba^{41}ba^{82}ba^{165}$.

Since $B_{\log{k}}[1] = w_k[1] = 1$, we have $2^{k+i-1} \leq |a^{w_k[1 : k+i]}| \leq 2^{k+i} - 1$ for $1 \leq i \leq k$ and thus $|s_k| \in \Theta(4^k)$.

Claim 1. A smallest SLP producing $s_k$ has size $O(k)$.

There is an SLP $A$ of size $O(k)$ for the first $a$-block $a^{w_k[1 : k+1]}$ of length $\Theta(2^k)$. Let $A$ be the start nonterminal of $A$. For the second $a$-block $a^{w_k[k+2]}$, we only need one additional rule: if $w_k[k+2] = 0$, then we can produce $a^{w_k[1 : k+2]}$ by the fresh nonterminal $B$ using the rule $B \rightarrow AA$. Otherwise, if $w_k[k+2] = 1$, then we use $B \rightarrow AAa$. The iteration of that process yields for each $a$-block only one additional rule of size at most 3. If we replace the $a$-blocks in $s_k$ by nonterminals as described, then the resulting word has size $2k + 1$ and hence $g(s_k) \in O(k)$.

Claim 2. The SLP produced by RePair on input $s_k$ has size $\Omega(k^2 / \log{k})$.

On unary inputs of length $m$, the start rule produced by RePair is strongly related to the binary encoding of $m$ as described above. On input $s_k$, the algorithm starts to produce a start rule which is similarly related to the binary words $w_k[1 : k + i]$ for $1 \leq i \leq k$. Consider the SLP $G$ which is produced by RePair after $(k-1)$ rounds on input $s_k$. We claim that up to this point RePair is not affected by the $b$’s in $s_k$ and therefore has introduced the rules $X_1 \rightarrow aa$ and $X_i \rightarrow X_{i-1}X_{i-1}$ for $2 \leq i \leq k-1$. If this is true, then the start rule after $k-1$ rounds begins with

$$S \rightarrow X_{k-1}X_{k-1}f_{k-1}(w_k[2])f_{k-2}(w_k[3]) \cdots f_0(w_k[k+1])b \cdots$$

where $f_0(1) = a$, $f_0(0) = \epsilon$ and $f_i(1) = X_i$, $f_i(0) = \epsilon$ for $i \geq 1$. All other $a$-blocks are longer than the first one, hence each factor of the start rule which corresponds to an $a$-block begins with $X_{k-1}X_{k-1}$. Therefore, the number of occurrences of $X_{k-1}X_{k-1}$ in the SLP is at least $k$. Since the symbol $b$ occurs only $k-1$ times in $s_k$, it follows that our assumption is correct and RePair is not affected by the $b$’s in the first $(k-1)$ rounds on input $s_k$. Also, for each block $a^{w_k[1 : k+i]}$, the $k-1$ least significant bits of $w_k[1 : k+i]$ $(1 \leq i \leq k)$ are represented in the corresponding factor of the start rule of $G$, i.e., the start rule contains non-overlapping factors $v_i$ with

$$v_i = f_{k-2}(w_k[i+2])f_{k-3}(w_k[i+3]) \cdots f_1(w_k[k+i-1])f_0(w_k[k+i]) \quad (1)$$

for $1 \leq i \leq k$. For example after 3 rounds on input $s_4 = a^{20}ba^{41}ba^{82}ba^{165}$, we have the start rule

$$S \rightarrow X_3X_3X_2bX_3a bX_3X_1bX_2a,$$

where $v_1 = X_2$, $v_2 = a$, $v_3 = X_1$ and $v_4 = X_2a$. The length of the factor $v_i \in \{a, X_1, \ldots, X_{k-2}\}^*$ from equation (1) is exactly the number of $1$’s in the word $w_k[i+2 : k+i]$. Since $w_k$ is constructed by the homomorphism $h$, it is easy to see that $|v_i| \geq (k-3)/2$. Note that no letter occurs more than once in $v_i$, hence $g(v_i) = |v_i|$. Further, each substring of length $2\lceil \log{k} \rceil + 2$ occurs
at most once in \(v_1, \ldots, v_k\), because otherwise there would be a factor of length \([\log k]\) occurring more than once in \(B_{[\log k]}\). It follows that there are at least
\[
k \cdot (\lceil (k - 3)/2 \rceil - 2\lceil \log k \rceil - 1) \in \Theta(k^2)
\]
different factors of length \(2[\log k] + 2 \in \Theta(\log k)\) in the right-hand side of the start rule of \(G\). By Lemma 1 it follows that a smallest SLP for the right-hand side of the start rule has size \(\Omega(k^2/\log k)\) and therefore \(|\text{RePair}(s_k)| \in \Omega(k^2/\log k)\).

In conclusion: We showed that a smallest SLP for \(s_k\) has size \(O(k)\), while RePair produces an SLP of size \(\Omega(k^2/\log k)\). This implies \(\alpha_{\text{RePair}}(s_k) \in \Omega(k^2/\log k)\), which together with \(n = |s_k|\) and \(k \in \Theta(\log n)\) finishes the proof. \(\square\)

Note that in the above prove, RePair chooses in the first \(k - 1\) rounds a digram for the replaced maximal string. Therefore, Theorem 2 also holds for the RePair-variant, where in every round a digram (which is not necessarily a maximal string) is replaced.

References

[1] J. Berstel and S. Brlek. On the length of word chains. *Inf. Process. Lett.*, 26(1):23–28, 1987.
[2] N. de Bruijn. A combinatorial problem. *Nederl. Akad. Wet., Proc.*, 49:758–764, 1946.
[3] M. Charikar, E. Lehman, A. Lehman, D. Liu, R. Panigrahy, M. Prabhakaran, A. Sahai, and A. Shelat. The smallest grammar problem. *IEEE Trans. Inf. Theory*, 51(7):2554–2576, 2005.
[4] F. Claude and G. Navarro. Fast and compact web graph representations. *ACM Transactions on the Web*, 4(4), 2010.
[5] A. A. Diwan. A new combinatorial complexity measure for languages. Tata Institute, Bombay, India, 1986.
[6] M. Gańczorz and A. Jeż. Improvements on re-pair grammar compressor. *to appear in Proceedings of DCC 2017*. IEEE Computer Society, 2017.
[7] R. González and G. Navarro. Compressed text indexes with fast locate. In *Proceedings of CPM 2007*, volume 4580 of *Lecture Notes in Computer Science*, pages 216–227. Springer, 2007.
[8] D. Hucke, M. Lohrey, and P. Reh. The smallest grammar problem revisited. *Proceedings of SPIRE 2016*, LNCS 9954, pages 35–49. Springer 2017.
[9] A. Jeż. Approximation of grammar-based compression via recompression. *Theoretical Computer Science*, 592:115–134, 2015.
[10] A. Jeż. A really simple approximation of smallest grammar. *Theoretical Computer Science*, 616:141–150, 2016.
[11] J. C. Kieffer and E.-H. Yang. Grammar-based codes: A new class of universal lossless source codes. *IEEE Trans. Inf. Theory*, 46(3):737–754, 2000.
[12] J. C. Kieffer, E.-H. Yang, G. J. Nelson, and P. C. Cosman. Universal lossless compression via multilevel pattern matching. *IEEE Trans. Inf. Theory*, 46(4):1227–1245, 2000.

[13] N. J. Larsson and A. Moffat. Offline dictionary-based compression. *Proceedings of DCC 1999*, pages 296–305. IEEE Computer Society, 1999.

[14] G. Navarro, S. J. Puglisi, and D. Valenzuela. Practical compressed document retrieval. In *Proceedings of SEA 2011*, volume 6630 of *Lecture Notes in Computer Science*, pages 193–205. Springer, 2011.

[15] G. Navarro and L. M. S. Russo. Re-pair achieves high-order entropy. In *Proceedings of DCC 2008*, page 537. IEEE Computer Society, 2008.

[16] C. G. Nevill-Manning and I. H. Witten. Identifying hierarchical structure in sequences: A linear-time algorithm. *J. Artif. Intell. Res.*, 7:67–82, 1997.

[17] W. Rytter. Application of Lempel-Ziv factorization to the approximation of grammar-based compression. *Theor. Comput. Sci.*, 302(1–3):211–222, 2003.

[18] J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE Trans. Inf. Theory*, 24(5):530–536, 1977.