Vol.2019-AL-171 No.6
2019/1/30

極大反復部分文字列に基づく文法圧縮

古谷 勇[1,a)]　髙木 拓也[1,b)]　中島 祐人[2,c)]　稲永 俊介[2,d)]　坂内 英夫[2,3,e)]　喜田 拓也[1,f)]

概要：本稿では，文法圧縮アルゴリズムのひとつである RePair を解析し，その生成文法と極大反復部分文字列（maximal repeat）との関係性を明らかにする．その上で，RePair を拡張した新たな文法圧縮アルゴリズムである MR-RePair を提案する．RePair が最頻出の文字ペアを順次変数に置換していくアルゴリズムであるのに対し，MR-RePair は最頻出の極大反復部分文字列を順次変数に置換していくものである．MR-RePair は RePair と同じデータ構造を用いて，入力文字列長に対して線形時間で実行可能である．最後に，RePair との比較実験の結果から，MR-RePair の圧縮性能の優位性を示す．

1. はじめに

文法圧縮は，入力文字列を一意に導出する文脈自由文法を構築することでデータサイズを削減する可逆圧縮の手法である．そうした文法のうちサイズが最小のものを求める問題は NP-hard である [5] が，これまでにいくつかの近似的なアルゴリズムが提案されてきた．中でも **RePair** [10] は，高い圧縮性能を発揮することで知られるアルゴリズムである [6], [8], [15]．実験的な成果のみならず，RePair は生成する文法のサイズについての理論的な解析研究も行われている [5], [12], [13].

近年，与えられた文字列がどの程度反復的（repetitive）であるかを測る指標として，**極大反復部分文字列（maximal repeat）** が注目されている．Belazzougui ら [3] は，Burrows-Wheeler 変換における連の数，および Lempel-Ziv 分割における要素数の上界が極大反復部分文字列の指標を用いて表せることを示した．また，データ構造分野では，極大反復部分文字列の拡張数でサイズが理論的にバウンドされる索引構造が提案されている [2], [14].

本稿では，極大反復部分文字列を基に RePair を解析する．上述の通り，先行研究として RePair の解析を行ったものはいくつか存在するが，我々の知る限り極大反復部分文字列と関係づけて研究した成果は存在しない．加えて，本稿ではこの極大反復部分文字列の性質を利用した文法圧縮アルゴリズム **MR-RePair** を提案する．近しい研究としては，（極大でない）反復部分文字列に着目した文法圧縮アルゴリズムがいくつか提案されている [1], [9], [11]．直近では，Gańczorz と Jeż が RePair の圧縮性能をヒューリスティックに改善することを試みた成果 [7] が存在する．しかしながら，これらはいずれも極大反復部分文字列の性質については論じていない．本稿では，ある条件のもとで MR-RePair が生成する文法のサイズが常に RePair の生成する文法のサイズ以下になることを理論的に示す．また，RePair と MR-RePair で比較実験を行い，MR-RePair の有効性を実験的に確認する．

本稿における主たる成果は以下の通りである．(1) RePair を解析し，その生成する文法と極大反復部分文字列との関係を示した．(2) RePair を拡張し，最頻出な極大反復部分文字列の置換に基づく新たなアルゴリズム MR-RePair を開発した．(3) MR-RePair を実装し RePair と比較実験を行い，生成文法サイズの観点から MR-RePair の有効性を実験的に確認した．

以降の構成は次の通りである．第 2 節では，文字列に関する記法の確認と，極大反復部分文字列，文法圧縮，RePair の定義を行う．第 3 節では RePair を解析し，極大反復部分文字列との関係を示す．第 4 節で MR-RePair を定義し，その実装方法についても述べる．第 5 節では RePair と MR-RePair の比較実験の結果を示す．最後に，第 6 節でまとめを行う．

[1] 北海道大学
Sapporo, Hokkaido 060–0814, Japan
[2] 九州大学
Fukuoka, Fukuoka 819–0395, Japan
[3] 理化学研究所 革新知能統合研究センター
Chūō, Tokyo 103–0027, Japan
[a)] furuya@ist.hokudai.ac.jp
[b)] tkg@ist.hokudai.ac.jp
[c)] yuto.nakashima@inf.kyushu-u.ac.jp
[d)] inenaga@inf.kyushu-u.ac.jp
[e)] bannai@inf.kyushu-u.ac.jp
[f)] kida@ist.hokudai.ac.jp

2. 準備

2.1 文字列に関する記法

記号の有限順序集合を**アルファベット**といい，Σ で表す．Σ^* の要素 $T = t_1 \cdots t_n$ を**文字列**といい，$|T| = n$ でその長さを表す．$T = t_1 \cdots t_n \in \Sigma^n$ は長さ n の文字列である．$T = usw$ が $u, s, w \in \Sigma^*$ とする文字列であるとき，s を T の**部分文字列**という．さらに，$1 \leq i \leq j \leq n$ であるような整数 i と j について，$T[i..j] = t_i \cdots t_j$ で，位置 i から j までの T の部分文字列を表し，$T[i] = t_i$ で T の i 番目の記号を表す．s を文字列としたとき，s が文字列 T にその部分文字列として出現する回数を s の**頻度**といい，$\#\mathrm{occ}_T(s)$ で表す．文脈から対象となる文字列 T を区別する必要がないときは，単に $\#\mathrm{occ}(s)$ と書く．文字列 Σ^* と $\hat{\Sigma}^*$ について，Σ から $\hat{\Sigma}$ への同型写像が存在するとき，両者をアルファベット Σ と $\hat{\Sigma}$ に関して**同型**であるという．

2.2 極大反復部分文字列

s を文字列 T の部分文字列とする．s の頻度が 1 より大きいとき，s を**反復部分文字列**という．$w \in \Sigma^*$ について，ws（または sw）の形で書けるすべての T の部分文字列の頻度が s の頻度より真に小さくなるとき，s を**左**（または**右**）**極大**であるといい，s が左極大かつ右極大であるとき，s を T の **極大反復部分文字列** という．例えば，$T =$ `abracadabra` の部分文字列 `abra` は極大反復部分文字列であるが，`br` は極大反復部分文字列ではない．

2.3 文法圧縮

V を**変数**の有限順序集合，Σ をアルファベット，R を**規則**と呼ばれる V と $(V \cup \Sigma)^*$ 間の二項関係の有限集合，$S \in V$ を**開始記号**と呼ばれる特別な変数としたとき，4 つ組 $G = \{V, \Sigma, S, R\}$ を**文脈自由文法**（CFG，または単に**文法**）という．規則は変数がどのように代入されるかを表し，$v \in V$ および $w \in (V \cup \Sigma)^*$ について $v \to w$ の形で書く．$X, Y \in V \cup \Sigma^*$ について，$X = x_l x x_r$, $Y = x_l y x_r$ かつ $x \to y \in R$ であるような $x_l, x, x_r, y \in (V \cup \Sigma)^*$ が存在するとき，$X \Rightarrow Y$ と書き，$\Rightarrow$ の反射的推移的閉包を $\overset{*}{\Rightarrow}$ で表す．$X \overset{*}{\Rightarrow} Y$ であるとき，Y は X から導出されるという．文法 G について，導出が常に一意であるとき，G は決定的であるという．$\hat{V} \subseteq V$ かつ $\hat{\Sigma} \subseteq V \cup \Sigma$ かつ $\hat{R} \subseteq R$ であるとき，$\hat{G} = \{\hat{V}, \hat{\Sigma}, \hat{S}, \hat{R}\}$ を G の**部分文法**という．

文法圧縮は，入力文字列 T に対し，それを一意に導出する制約付きの文脈自由文法を構築する可逆圧縮手法である．G が決定的であるために，各 $v \in V$ を左辺とする規則は R 中に唯一つである必要がある．以降，文法はすべて決定的であるとし，各規則は $v_i \to expr_i$ の形をとるものとする．ここで，$expr_i$ は $expr_i = a$ $(a \in \Sigma)$ または $expr_i = v_{j_1} v_{j_2} \cdots v_{j_n}$ ($i > j_k$ ただし $1 \leq k \leq n$) である．以降，文法圧縮に関連する説明の中で，変数 V およびアルファベット Σ から成る系列 $(V \cup \Sigma)^*$ を文字列と呼ぶことがある．

文法圧縮においては，構築した文法サイズを指標として圧縮性能を評価する．文法サイズは，構築した文法のすべての規則の右辺の文字数の合計である．

2.4 RePair

RePair は Larsson と Moffat [10] によって提案された文法圧縮アルゴリズムである．入力文字列 T に対し，RePair は以下の方法で文法 $G = \{V, \Sigma, S, R\}$ を構築する：

Step 1. 各記号 $a \in \Sigma$ を新たな変数 v_a に置換し，$v_a \to a$ を R に追加する．

Step 2. T 中の最頻出な 2 文字から成る部分文字列（ペア）p を見つける．

Step 3. T 中の p のすべての出現を新たな変数 v で（p が重複して出現する場合は可能な限り）置換し，$v \to p$ を R に追加する．

Step 4. **Step 3** で生成された文字列に対して各ペアの頻度を数え直す．最頻出ペアの頻度が 1 ならば，$S \to$ (現在の文字列) を R に追加して終了する．そうでない場合は，**Step 2** に戻る．

図 1 に RePair の文法構築過程の動作例を示す．

補題 1 ([10])**.** RePair は，Word RAM モデルにおいて，$\mathcal{O}(n)$ 時間，$5n + 4k^2 + 4k' + \lceil\sqrt{n+1}\rceil - 1$ ワードサイズで動作する．ここで，n は入力文字列長，k は入力文字列のアルファベットサイズ，k' は最終的な辞書サイズである．

3. RePair の解析

本節では，RePair の極大反復部分文字列との関係性の観点からの解析を行う．

次の定理は，RePair は再帰的に最頻出な極大反復部分文字列を置換するという，RePair の挙動な本質的性質を示したものである．

定理 1. T を入力文字列とし，T 中のいかなる最頻出な極大反復部分文字列も，自身と重複するような出現を持たないとする．f を T 中の最頻出ペアの頻度，t を T 中のすべての頻度 f のペアを変数に置換した後に得られる文字列とする．このとき，T 中のすべての頻度 f の極大反復部分文字列を変数に置換して得られる文字列のうち，t と同型な文字列 s が存在する．

定理 1 を証明するために，2 つの補題と 1 つの系が必要である．以下の定理は，文字列中の最頻出な極大反復部分文字列と最頻出なペアとの基礎的な関係性を示したものである．尚，紙面の都合上，以降度々証明を省略することが

	a	b	r	a	c	a	d	a	b	r	a
$v_\alpha \to \alpha$ $(\alpha = \mathtt{a},\mathtt{b},\mathtt{r},\mathtt{c},\mathtt{d})$	$v_\mathtt{a}$	$v_\mathtt{b}$	$v_\mathtt{r}$	$v_\mathtt{a}$	$v_\mathtt{c}$	$v_\mathtt{a}$	$v_\mathtt{d}$	$v_\mathtt{a}$	$v_\mathtt{b}$	$v_\mathtt{r}$	$v_\mathtt{a}$
$v_1 \to v_\mathtt{a}v_\mathtt{b}$	v_1		$v_\mathtt{r}$	$v_\mathtt{a}$	$v_\mathtt{c}$	$v_\mathtt{a}$	$v_\mathtt{d}$	v_1		$v_\mathtt{r}$	$v_\mathtt{a}$
$v_2 \to v_1v_\mathtt{r}$	v_2			$v_\mathtt{a}$	$v_\mathtt{c}$	$v_\mathtt{a}$	$v_\mathtt{d}$	v_2			$v_\mathtt{a}$
$v_3 \to v_2v_\mathtt{a}$	v_3				$v_\mathtt{c}$	$v_\mathtt{a}$	$v_\mathtt{d}$	v_3			
$S \to v_3v_\mathtt{c}v_\mathtt{a}v_\mathtt{d}v_3$	S										

図 1 文字列 abracadabra に対する RePair の動作過程の例．構築された文法は $\{\{v_\mathtt{a}, v_\mathtt{b}, v_\mathtt{r}, v_\mathtt{c}, v_\mathtt{d}, v_1, v_2, v_3, S\}, \{\mathtt{a},\mathtt{b},\mathtt{r},\mathtt{c},\mathtt{d}\}, S, \{v_\mathtt{a} \to \mathtt{a}, v_\mathtt{b} \to \mathtt{b}, v_\mathtt{r} \to \mathtt{r}, v_\mathtt{c} \to \mathtt{c}, v_\mathtt{d} \to \mathtt{d}, v_1 \to v_\mathtt{a}v_\mathtt{b}, v_2 \to v_1v_\mathtt{c}, v_3 \to v_2v_\mathtt{d}, S \to v_3v_\mathtt{c}v_\mathtt{a}v_\mathtt{d}v_3\}\}$ で，サイズは 16 である．

ある．

補題 2. ペア p が文字列 T 中で最頻出であることと，p が T の最頻出な極大反復部分文字列中に 1 度だけ出現することは同値である．

以下の系は，補題 2 より直ちに導かれる．

系 1. 文字列中の最頻出なペアの頻度と最頻出な極大反復部分文字列の頻度は等しい．

以下の補題は，極大反復部分文字列の重要な性質を示したものである．

補題 3. 最頻出な極大反復部分文字列同士の重複部分の長さは高々 1 である．

上記の補題と系より，定理 1 が証明できる．

証明 (定理 1 の証明)．系 1 より，T 中の最頻出な極大反復部分文字列の頻度は f である．T 中の最頻出なペアの 1 つを p とする．補題 2 より，p を 1 度だけ含む最頻出な極大反復部分文字列が唯一つ存在し，いまこれを r とする．また，いま T 中には $z, w \in \Sigma$，$x, y \in \Sigma^*$，$xpy = r$ であるような部分文字列 $zxpyw$ が存在すると仮定し，$r[1]$ と $r[|r|]$ をそれぞれ $\dot{x}$ および $\dot{y}$ と書く．このとき，以下の 2 通りの場合が考えられる：

(i) $\#\mathrm{occ}(z\dot{x}) < f$ **かつ** $\#\mathrm{occ}(\dot{y}w) < f$ **のとき．** $|r| = 2$ ならば，$p = r$ なので，p の置換はすなわち最頻出な極大反復部分文字列の置換である．

$|r| > 2$ の場合は，p が変数 v に置換された後，r は xvy となる．これは更新後の文字列中に f 回出現し，補題 2 より，xvy 中に出現する各ペアの頻度はすべて f である．ペアの最頻頻度は増加しないため，置換後も f が変わらず最頻頻度である．したがって，その後の過程において xvy に含まれるすべてのペアは再帰的に変数に置換され，$z\dot{x}$ と $\dot{y}w$ はそれまで置換されないままである．以上は p のすべての出現に関して成り立ち，最頻頻度が変わらない間のペアの置換が，最頻出な極大反復部分文字列中のペアの置換に対応することがわかる．よって，r の置換により s が得られる．

(ii) $\#\mathrm{occ}(z\dot{x}) = f$ **または** $\#\mathrm{occ}(\dot{y}w) = f$ **のとき．** $\#\mathrm{occ}(z\dot{x}) = f$ の場合を考える．xpy が極大反復部分文字列であるという仮定より，$\#\mathrm{occ}(zxpy) < f$ である．いま，RePair において p よりも先に $z\dot{x}$ が変数 v に置換されたとする．補題 2 より，$z\dot{x}$ を 1 度だけ含み，かつ f 回出現し，$r' \neq r$ であるような極大反復部分文字列 r' が存在する．補題 3 より，r と r' の重複部分の長さは高々 1 であるので，r と r' の両方に含まれる記号は x のみである．この時点で，$xpy = r$ はそのいくつかの出現が $vr[2..|r|]$ に変化しているため，最頻出な極大反復部分文字列ではなくなっている．しかしながら，$r[2..|r|]$ は更新後の文字列において f 回出現する．$\#\mathrm{occ}(zxpy) < f$ かつ $\#\mathrm{occ}(xpy) = f$ であるため，$\#\mathrm{occ}(vr[2]) < f$ かつ $r[2..|r|]$ は極大反復部分文字列である．よって **(i)** と同様に，$r[2..|r|]$ はその後の過程で 1 つの変数に置換される．このとき，r' も 1 つの変数になっている．したがって，はじめに r' を置換した後，$r[2..|r|]$ を置換することで，s が得られる．以上は，$\#\mathrm{occ}(\dot{y}w) = f$ の場合，および $\#\mathrm{occ}(z\dot{x}) = \#\mathrm{occ}(\dot{y}w) = f$ にも $\dot{y}w$ について同様に成り立つ． □

定理 1 より，現状の文字列に対して最頻出な極大反復部分文字列が 1 つしかないのであれば，そのすべての出現は RePair によって段階的に変数に置換される．しかしながら，最頻出な極大反復部分文字列が 2 つ以上存在し，それらの出現に重複がある場合は，単純にそうとは言い切れない．そうした場合においては，どの極大反復部分文字列が最初に置換され終わって 1 つの変数になるかという問題は最頻出なペアの置換順序に依存する．しかし，複数の最頻出なペアが存在するときの置換順序は，アルゴリズムの実装によって異なる．このような実装に依存する極大反復部分文字列の選択（置換）順序を **MR-order** と呼ぶ．

2 つ以上の最頻出なペアの出現に重複がある場合，RePair によって構築される文法のサイズはそれらの選択順に依存する．例えば，文字列 bcxdabcyabzdabvbcuda では，ab，bc，da の 3 つが最頻頻度 3 のペアである．RePair が最初に ab を置換したとすると，生成される文法の規則は $\{v_1 \to \mathtt{ab},\ v_2 \to \mathtt{bc},\ v_3 \to \mathtt{d}v_1,\ S \to v_2\mathtt{x}v_3\mathtt{cy}v_1\mathtt{z}v_3\mathtt{v}v_2\mathtt{uda}\}$ のようになり，このときの文法サイズは 19 である．一方で，もし RePair が最初に da を置換したとすると，生成される文法の規則は $\{v_1 \to \mathtt{da},\ v_2 \to \mathtt{bc},\ S \to v_2\mathtt{x}v_1v_2\mathtt{yabz}v_1\mathtt{bv}v_2\mathtt{u}v_1\}$

となり，このときの文法サイズは 18 である．

注意 1. 最頻出な異なるペアが 2 種類以上存在するとき，RePair が構築する文法のサイズは，それらを置換する順序に依存する．

しかしながら，以下の定理は，ペアの置換順序よりも MR-order の方が，RePair の構築する文法のサイズを決定する上では本質的に重要であることを示している．

定理 2. 同じ MR-order に従って動作するならば，RePair の構築する文法のサイズは常に等しい．

証明. T を RePair の文法構築過程に現れる文字列とし，f を T 中の最頻出なペアの頻度とする．また，T' を T 中のすべての頻度 f のペアを置換し終えて得られる文字列とする．定理 1 より，T' として考え得るすべての文字列は，ペアの置換順序に拘らず互いに同型であり，その長さは等しい．いま考える MR-order において最も優先される T 中の最頻出な極大反復部分文字列を r_1 とする．r_1 はすべての頻度 f のペアが置換され終わった時点で 1 つの変数に変換されており，補題 2 より，r_1 中に出現するペアはすべて互いに異なる．したがって，ちょうど r_1 のみを導出するような部分文法のサイズは $2(|r_1|-1)+1=2|r_1|-1$ である．これは，次に優先度の高い極大反復部分文字列（r_2 で表す）についても同様に成り立つ．r_2 の出現に r_1 の出現との重複が存在する場合，実際に 1 つの変数にまとめあげられるのは，r_2 の両端のいずれかの変数を除いた部分文字列になる．しかしながら，同じ MR-order においては，このような部分文字列はペアの置換順序によらず一定であり，これを導出する部分文法のサイズも変わらない．以上は，RePair の全体の動作を通じてすべての最頻出な極大反復部分文字列に同様に成り立つ． □

4. MR-RePair

はじめに提案手法の前身となる Naïve-MR-RePair アルゴリズムについて述べる．これは元の RePair よりも大きいサイズの文法を出力する場合もあるが，単純であり，提案手法の理解の助けになる．その後，提案手法である MR-RePair を解説する．

定義 1 (Naïve-MR-RePair)**.** 入力文字列 T に対して，Naïve-MR-RePair が生成する文法を $G=\{V,\Sigma,S,R\}$ とする．このとき，Naïve-MR-RePair は以下の手順で G を構築する：

Step 1. 各記号 $a\in\Sigma$ を新たな変数 v_a に置換し，$v_a\to a$ を R に追加する．

Step 2. T 中の最頻出な 2 文字以上の極大反復部分文字列 r を見つける．

Step 3. T 中の r のすべての出現を新たな変数 v で（r が重複して出現する場合は可能な限り）置換し，$v\to r$ を R に追加する．

Step 4. **Step 3** で生成された文字列に対して各極大反復部分文字列の頻度を数え直す．最頻出な極大反復部分文字列の頻度が 1 ならば，$S\to$ (現在の文字列) を R に追加して終了する．そうでない場合は，**Step 2** に戻る．

前述した MR-order の概念は，この Naïve-MR-RePair にも簡単に適用可能である．

Naïve-MR-RePair の文法構築過程の例を図 2 に示す．図 1 と図 2 を比較すると，ペアよりも極大反復部分文字列を用いる方が有効であることの直感的な理由が見て取れる．文字列 $v_{\mathtt{a}}v_{\mathtt{b}}v_{\mathtt{r}}v_{\mathtt{a}}v_{\mathtt{c}}v_{\mathtt{a}}v_{\mathtt{d}}v_{\mathtt{a}}v_{\mathtt{b}}v_{\mathtt{r}}v_{\mathtt{a}}$ に対して動作する際，RePair と Naïve-MR-RePair はともに最頻出な極大反復部分文字列である $v_{\mathtt{a}}v_{\mathtt{b}}v_{\mathtt{r}}v_{\mathtt{a}}$ を導出する部分文法を構築する．この部分文法の規則は，RePair では $\{v_1\to v_{\mathtt{a}}v_{\mathtt{b}},v_2\to v_1v_{\mathtt{r}},v_3\to v_2v_{\mathtt{a}}\}$ となり，そのサイズは 6 であるが，Naïve-MR-RePair では $\{v_1\to v_{\mathtt{a}}v_{\mathtt{b}}v_{\mathtt{r}}v_{\mathtt{a}}\}$ であり，サイズは 4 である．

しかしながら，次の定理は，同じ MR-order で動作する Naïve-MR-RePair と RePair で比較しても，特定の状況においては Naïve-MR-RePair で構築した文法が RePair によるそれよりもサイズが大きくなることを示している．

定理 3. 長さ n の文字列 T に対して，RePair と Naïve-MR-RePair が同じ MR-order で動作すると仮定する．g_{rp} および g_{nmr} を，それぞれ RePair と Naïve-MR-RePair が T に対して構築した文法のサイズとする．このとき，$g_{nmr}=g_{rp}+\mathcal{O}(\log n)$ となるような場合が存在する．

証明. RePair および Naïve-MR-RePair によって構築された文法を，それぞれ $G_{rp}=\{V_{rp},\Sigma_{rp},S_{rp},R_{rp}\}$，$G_{nmr}=\{V_{nmr},\Sigma_{nmr},S_{nmr},R_{nmr}\}$ とする．$T'=v_1\cdots v_n$ を $v_i\in V_{rp}\cap V_{nmr}$，$v_i\to T[i]\in R_{rp}\cap R_{nmr}$ $(i=1,\cdots,n)$ であるような文字列とし，$\hat{G}_{rp}=\{\hat{V}_{rp},\hat{\Sigma}_{rp},\hat{S}_{rp},\hat{R}_{rp}\}$（または $\hat{G}_{nmr}=\{\hat{V}_{nmr},\hat{\Sigma}_{nmr},\hat{S}_{nmr},\hat{R}_{nmr}\}$）を，$T'$ をちょうど導出するような G_{rp}（または G_{nmr}）の部分文法とする．いま，$T'=(uw)^{2^{m+1}-1}u$ を $u\in V_{rp}\cap V_{nmr}$，$w\in(V_{rp}\cap V_{nmr})^+$，$m\in\mathbb{N}^+$，かつ uwu が T' の最頻出な極大反復部分文字列であるような文字列とする．ここで，$2^{m+1}-1=\sum_{i=0}^{m}2^i$ である．このとき，$\hat{R}_{rp}$ と $\hat{R}_{nmr}$ 以下のようになる．

$\hat{R}_{rp}$**:** $x_i\in\hat{V}_{rp}$ $(1\le i\le m)$ および $y_j\in\hat{V}_{rp}\cup\hat{\Sigma}_{rp}$ $(1\le j\le|w|)$ とする．このとき，$\hat{R}_{rp}$ は次の規則からなる．

- $|w|$ 個の $y_j\to y_l y_r$（ただし，$y_{|w|}\overset{*}{\Rightarrow}uw$），
- 1 つの $x_1\to y_{|w|}y_{|w|}$ と $\log_2\lfloor 2^{m+1}-1\rfloor-1=m-1$ 個の $x_i\to x_{i-1}x_{i-1}$ $(2\le i\le m)$，
- 1 つの $\hat{S}_{rp}\to x_m x_{m-1}\cdots x_1 y_{|w|}$.

$\hat{R}_{nmr}$**:** $d=|\hat{V}_{nmr}|=|\hat{R}_{nmr}|$ および $z_i\in\hat{V}_{nmr}$ $(1\le i\le d)$ とする．このとき，$\hat{R}_{nmr}$ は次の規則からなる．

- 1 つの $z_1\to uwu$，
- $d-1$ 個の $z_i\to z_{i-1}wz_{i-1}$（ただし，$2\le i\le d$ かつ

	a	b	r	a	c	a	d	a	b	r	a
$v_\alpha \to \alpha$ $(\alpha = \mathtt{a},\mathtt{b},\mathtt{r},\mathtt{c},\mathtt{d})$	$v_\mathtt{a}$	$v_\mathtt{b}$	$v_\mathtt{r}$	$v_\mathtt{a}$	$v_\mathtt{c}$	$v_\mathtt{a}$	$v_\mathtt{d}$	$v_\mathtt{a}$	$v_\mathtt{b}$	$v_\mathtt{r}$	$v_\mathtt{a}$
$v_1 \to v_\mathtt{a}v_\mathtt{b}v_\mathtt{r}v_\mathtt{a}$	v_1				$v_\mathtt{c}$	$v_\mathtt{a}$	$v_\mathtt{d}$	v_1			
$S \to v_1v_\mathtt{c}v_\mathtt{a}v_\mathtt{d}v_1$	S										

図 2 文字列 abracadabra に対する Naïve-MR-RePair の動作過程の例．構築された文法は $\{\{v_\mathtt{a}, v_\mathtt{b}, v_\mathtt{r}, v_\mathtt{c}, v_\mathtt{d}, v_1, S\}, \{\mathtt{a},\mathtt{b},\mathtt{r},\mathtt{c},\mathtt{d}\}, S, \{v_\mathtt{a} \to \mathtt{a}, v_\mathtt{b} \to \mathtt{b}, v_\mathtt{r} \to \mathtt{r}, v_\mathtt{c} \to \mathtt{c}, v_\mathtt{d} \to \mathtt{d}, v_1 \to v_\mathtt{a}v_\mathtt{b}v_\mathtt{r}v_\mathtt{a}, S \to v_1v_\mathtt{c}v_\mathtt{a}v_\mathtt{d}v_1\}\}$ で，サイズは 14 である．

$z_d = \hat{S}_{nmr}$)．
$\hat{g}_{rp}$ と $\hat{g}_{nmr}$ を，それぞれ $\hat{G}_{rp}$ および $\hat{G}_{nmr}$ のサイズとする．以上より，次が成り立つ．

$$\hat{g}_{rp} = 2|w| + 2m + (m+2) = 3m + 2|w| + 2 \quad (1)$$

$$\hat{g}_{nmr} = |w| + 2 + (|w|+2)(d-1) = (|w|+2)d \quad (2)$$

ここで，T' の長さについて以下が成り立つ．

$$(2^d - 1)|w| + 2^d = n = (2(2^m - 1) + 1)(|w| + 1) + 1.$$

右辺は $2^{m+1}|w| + 2^{m+1}$ となり，$d = m+1$．よって，式 (1) および 式 (2) より，

$$\hat{g}_{nmr} - \hat{g}_{rp} = (m-1)(|w| - 1) - 1$$

が成り立つ．したがって，$\hat{g}_{nmr} > \hat{g}_{rp}$ が成り立つような $(m, |w|)$ が存在する．□

定理 3 にあるように，Naïve-MR-RePair の生成文法が RePair のそれよりもサイズが大きくなってしまう理由は，最頻出な極大反復部分文字列の出現に自身の別の出現との重複が存在する場合，Naïve-MR-RePair はそのすべての出現を置換できないからである．以降では，Naïve-MR-RePair のこの点を改善したアルゴリズム MR-RePair を説明する．

定義 2 (MR-RePair)．入力文字列 T に対して，MR-RePair が生成する文法を $G = \{V, \Sigma, S, R\}$ とする．このとき，MR-RePair は以下の手順で G を構築する：
Step 1. 各記号 $a \in \Sigma$ を新たな変数 v_a に置換し，$v_a \to a$ を R に追加する．
Step 2. T 中の最頻出な 2 文字以上の極大反復部分文字列 r を見つける．
Step 3. $|r| > 2$ かつ $r[1] = r[|r|]$ が成り立つか確認し，もし成り立つならば，$r[2..|r|]$ を r と見なす．
Step 4. T 中の r のすべての出現を新たな変数 v で（r が重複して出現する場合は可能な限り）置換し， $v \to r$ を R に追加する．
Step 5. Step 3 で生成された文字列に対して各極大反復部分文字列の頻度を数え直す．最頻出な極大反復部分文字列の頻度が 1 ならば，$S \to$ (現在の文字列) を R に追加して終了する．そうでない場合は，**Step 2** に戻る．

MR-order の概念は，Naïve-MR-RePair と同様，MR-RePair にも適用できる．MR-RePair の文法構築過程の動作例を図 3 に示す．**Step 3** において，$r[2..|r|]$ の代わりに $r[1..|r|-1]$ としても問題ない．補題 3 より，最頻出な極大反復部分文字列の重複部分の長さは高々 1 であるので，いま MR-RePair は r のいくつかの出現に自身との重複があっても，そのすべての出現を変数に置換することができる．仮に，r が重複して出現していないにも拘らず $r[1] = r[|r|]$ であった場合でも，$r[2..|r|]$ が変数 v に置換された後，$r[1]v$ は最頻出な極大反復部分文字列であり，直ちに置換される．$|r| = 2$ の場合，その出現に重複があると，MR-RePair はそのすべてを変数に置換することができないが，これは RePair に関しても同様である．

定理 4. 長さ n の文字列 T に対して，RePair と MR-RePair が同じ MR-order で動作すると仮定する．g_{rp} および g_{mr} を，それぞれ RePair と MR-RePair が T に対して構築した文法のサイズとする．このとき，$\frac{1}{2}g_{rp} < g_{mr} \leq g_{rp}$ が成り立つ．

証明. RePair および MR-RePair によって構築された文法を，それぞれ $G_{rp} = \{V_{rp}, \Sigma_{rp}, S_{rp}, R_{rp}\}$，$G_{mr} = \{V_{mr}, \Sigma_{mr}, S_{mr}, R_{mr}\}$ とする．$T' = v_1 \cdots v_n$ を $v_i \in V_{rp} \cap V_{mr}$，$v_i \to T[i] \in R_{rp} \cap R_{mr}$（$i = 1, \cdots, n$）であるような文字列とする．

はじめに T' を考える．T' 中の最頻出な極大反復部分文字列の頻度を f_1 とする．いま，系 1 より，T' 中の最頻出なペアの頻度も f_1 である．RePair （または MR-RePair ）が頻度 f_1 のペア（または極大反復部分文字列）を置換している期間に生成する G_{rp}（または G_{mr}）の部分文法を $\hat{G}_{rp}^{(f_1)}$（または $\hat{G}_{mr}^{(f_1)}$）とし，$\hat{g}_{rp}^{(f_1)}$（または $\hat{g}_{mr}^{(f_1)}$）をそれらのサイズ，$T_{rp}^{(f_1)}$（または $T_{mr}^{(f_1)}$）を頻度 f_1 のペア（または極大反復部分文字列）のすべての出現を置換した後に得られる文字列とする．T' 中の頻度 f_1 の各極大反復部分文字列を $r_1^{(f_1)}, \cdots, r_{m_1}^{(f_1)}$ とし，これらが MR-order によってこの順で優先的であるとする．$\hat{G}_{rp}^{(f_1)}$ と $\hat{G}_{mr}^{(f_1)}$ のどちらにも，それを導出する変数が存在するような $r_i^{(f_1)}$ の最長部分文字列の長さをそれぞれ $l_i^{(f_1)}$（$i = 1, \cdots, m_1$）とする．このような部分文字列は RePair と MR-RePair で共通であり，各 $l_i^{(f_1)}$ は少なくとも 2 以上である．このとき，補題 2 より，以下が成り立つ．

	a	b	r	a	c	a	d	a	b	r	a
$v_\alpha \to \alpha$ $(\alpha = \mathtt{a},\mathtt{b},\mathtt{r},\mathtt{c},\mathtt{d})$	$v_\mathtt{a}$	$v_\mathtt{b}$	$v_\mathtt{r}$	$v_\mathtt{a}$	$v_\mathtt{c}$	$v_\mathtt{a}$	$v_\mathtt{d}$	$v_\mathtt{a}$	$v_\mathtt{b}$	$v_\mathtt{r}$	$v_\mathtt{a}$
$v_1 \to v_\mathtt{a}v_\mathtt{b}v_\mathtt{r}$	v_1			$v_\mathtt{a}$	$v_\mathtt{c}$	$v_\mathtt{a}$	$v_\mathtt{d}$	v_1			$v_\mathtt{a}$
$v_2 \to v_1v_\mathtt{a}$	v_2				$v_\mathtt{c}$	$v_\mathtt{a}$	$v_\mathtt{d}$	v_2			
$S \to v_2v_\mathtt{c}v_\mathtt{a}v_\mathtt{d}v_2$	S										

図 3 文字列 abracadabra に対する MR-RePair の動作過程の例．構築された文法は $\{\{v_\mathtt{a}, v_\mathtt{b}, v_\mathtt{r}, v_\mathtt{c}, v_\mathtt{d}, v_1, S\}, \{\mathtt{a},\mathtt{b},\mathtt{r},\mathtt{c},\mathtt{d}\}, S, \{v_\mathtt{a} \to \mathtt{a}, v_\mathtt{b} \to \mathtt{b}, v_\mathtt{r} \to \mathtt{r}, v_\mathtt{c} \to \mathtt{c}, v_\mathtt{d} \to \mathtt{d}, v_1 \to v_\mathtt{a}v_\mathtt{b}v_\mathtt{r}, v_2 \to v_1v_\mathtt{a}, S \to v_2v_\mathtt{c}v_\mathtt{a}v_\mathtt{d}v_2\}\}$, で，サイズは 15 である．

$$\hat{g}_{rp}^{(f_1)} = \sum_{i=1}^{m_1} 2(l_i^{(f_1)} - 1)\ , \quad \hat{g}_{mr}^{(f_1)} = \sum_{i=1}^{m_1} l_i^{(f_1)}\ ,$$
$$\therefore\ \frac{1}{2}\hat{g}_{rp}^{(f_1)} < \hat{g}_{mr}^{(f_1)} \le \hat{g}_{rp}^{(f_1)}. \tag{3}$$

更新後の文字列 $T_{rp}^{(f_1)}$ および $T_{mr}^{(f_1)}$ は，V_{rp} と V_{mr} について同型である．f_2 を $T_{rp}^{(f_1)}$ 中の最頻出な極大反復部分文字列の頻度とする（これは $T_{mr}^{(f_1)}$ 中でも同じである）．ここで，$\hat{G}_{rp}^{(f_2)}$ と $\hat{G}_{mr}^{(f_2)}$ に関しても以上と同様の議論が成り立つ．したがって，式 (3) と同様にして $\frac{1}{2}\hat{g}_{rp}^{(f_2)} < \hat{g}_{mr}^{(f_2)} \le \hat{g}_{rp}^{(f_2)}$ が成り立ち，更新後の文字列 $T_{rp}^{(f_2)}$ および $T_{mr}^{(f_2)}$ は同型である．帰納的に，すべての最頻頻度 f_i に関して $\frac{1}{2}\hat{g}_{rp}^{(f_i)} < \hat{g}_{mr}^{(f_i)} \le \hat{g}_{rp}^{(f_i)}$ が成り立ち，$T_{rp}^{(f_i)}$ と $T_{mr}^{(f_i)}$ は同型となる．k を $f_k > 1$ かつ $f_{k+1} = 1$ であるような自然数とし，これを RePair と MR-RePair の全体の動作過程において，最頻頻度が減少する回数とする．すると，

$$\begin{aligned} g_{rp} &= \sum_{j=1}^{k} \hat{g}_{rp}^{(f_j)} + |\Sigma| + |T_{rp}^{(f_k)}| \\ &= \sum_{j=1}^{k}\sum_{i=1}^{m_j} 2(l_i^{(f_j)} - 1) + |\Sigma| + |T_{rp}^{(f_k)}|\ , \end{aligned} \tag{4}$$

$$\begin{aligned} g_{mr} &= \sum_{j=1}^{k} \hat{g}_{mr}^{(f_j)} + |\Sigma| + |T_{mr}^{(f_k)}| \\ &= \sum_{j=1}^{k}\sum_{i=1}^{m_j} l_i^{(f_j)} + |\Sigma| + |T_{mr}^{(f_k)}| \end{aligned} \tag{5}$$

が成り立つ．すべての $l_i^{(f_j)} \ge 2$ と $|T_{rp}^{(f_k)}| = |T_{mr}^{(f_k)}|$ について，式 (4) および式 (5) より，$\frac{1}{2}g_{rp} < g_{mr} \le g_{rp}$ である．$g_{mr} = g_{rp}$ は，すべての $l_i^{(f_j)}$ が 2 の場合に成り立つ． □

RePair と MR-RePair の MR-order が異なる場合，理論的には，MR-RePair の構築文法が RePair の構築文法より大きなサイズになることがある．この事実は，注意 1 と定理 4 より導かれる．

MR-RePair は [10] に述べられている RePair の実装を拡張することで，これと同じ計算量で実装可能である．

定理 5. 長さ n の文字列に対して MR-RePair が構築する文法を $G = \{V, \Sigma, S, R\}$ とする．このとき，MR-RePair は $\mathcal{O}(n)$ 時間，$5n + 4k^2 + 4k' + \lceil\sqrt{n+1}\rceil - 1$ ワードサイズで動作する．ここで，k と k' はそれぞれ Σ と V の大きさである．

証明. RePair と比較して，MR-RePair の実装において必要となる付加的な操作には次の 2 つがある．(i) 選択した最頻出なペアを，それが極大反復部分文字列である限り左右に伸張していく．(ii) 得られた長さ 3 以上の極大反復部分文字列についてその両端の文字を確認し，それらが同じ場合には一方をパターンから取り除く．これらは RePair で用いられるものとまったく同じデータ構造で実現可能である．よって，MR-RePair の領域計算量は補題 1 に従う．

操作 (ii) は明らかに定数時間で実行可能であるので，操作 (i) について考える．いま，選択した最頻出なペアを含む最頻出な極大反復部分文字列の長さを l とし，その頻度を f とする．選択したペアのすべての出現についてその左右拡張を調べ，自身が極大反復部分文字列になるまで伸張していく操作には $\mathcal{O}(fl)$ 時間必要である．しかし，これを変数に置換することで，全体の文字列は少なくとも $f(l-1)$ 文字短くなる．したがって，アルゴリズム全体での置換可能回数より，MR-RePair は $\mathcal{O}(n)$ 時間で動作する． □

注意 2. RePair の文法は，規則の右辺にただ一度のみ出現するような変数を探し，その変数を左辺とする規則の右辺をそこに埋め込んで，余分な規則を削除する，という操作を繰り返すことで，簡単に MR-RePair の文法に変換できると思われるかもしれない．しかし，このような方法では，そうした変数の探索や記憶のために余計な操作やメモリが必要になるため，定理 5 に示すような時間領域計算量は達成できない．

5. 実験

MR-RePair を実装し，RePair および Gańczorz と Jeż [7] によって提案された Re-PairImp[*1] と，構築した文法のサイズと実行時間を測定した．

注意 1 で述べた通り，生成された文法のサイズは MR-order に依存する．実際には，MR-order はペアの頻度を管理する優先度つきキューの実装によって異なる．この点を観察するため，比較の際，Maruyama[*2]，Navarro[*3]，

[*1] https://bitbucket.org/IguanaBen/repairimproved
[*2] https://code.google.com/archive/p/re-pair/
[*3] https://www.dcc.uchile.cl/~gnavarro/software/index.html

Prezza[*4] [4]，Wan[*5]，Yoshida[*6] によって作成された 5 種類の RePair の実装を用いた．

表 1 は実験に用いた文字列データの詳細を示したものである．ランダム生成した文字列（rand77.txt），および Pizza&Chili Corpus[*7] の Repetitive Corpus より取得した，フィボナッチ文字列（fib41），ドイツ語の文字列データ（einstein.de.txt）の 3 つの文字列データを，繰り返しの多い文字列として採用した．rand77.txt は，英数文字といくつかの特殊記号から成り，1 行 63 文字（末尾の改行を含めると 64 文字）のランダムな文字列を 1024 行含んだブロックを 32 回コピーして連結したものである．よって，サイズは $64 \times 1024 \times 32 = 2,097,152$ byte である．さらに，Large Corpus[*8] より 3 つのデータ（E.coli, bibile.txt, world192.txt）を実データとして採用した．各プログラムは各データについて 7 回実行し，文法構築処理についてのみ経過 CPU 時間を計測した．その上で，最大値と最小値を除いた 5 つの結果に対して平均を算出した．実験は Intel(R) Xeon(R) E5-2670 2.30GHz dual CPU with 64GB RAM，Ubuntu 16.04LTS on Windows 10 上で行った．すべてのプログラムは gcc version 7.3.0 にて，“-O3” オプションつきでコンパイルしている．

表 2 は実験結果をまとめたものである．ここでは，MR-RePair でも RePair でも同じ数であるため，単一の終端記号を導出する規則は数に含めていない．表から読み取れる通り，fib41 以外のすべてのデータに対して，RePair の文法サイズは実装によって異なっている．またすべてのデータについて，MR-RePair は RePair よりサイズが小さい文法を生成することができた．特に rand77.txt に関しては，規則数は約 11%，文法サイズは約 55%に減少している．さらに einstein.de.txt に関しては，規則数は約 44%，文法サイズは約 72%に減少している．一方で，特別に繰り返しの多くない Large Corpus のデータに関しては，MR-RePair による圧縮性能の改善は限定的である．fib41 は，アルゴリズムの動作のいずれの段階においても，長さが 2 より大きく重複しない最頻出の極大反復部分文字列が出現しない．したがって，MR-RePair は RePair とまったく同じ文法を生成する．また，MR-RePair の速度は RePair の最も速い実装とほぼ変わらないことがわかった．

6. おわりに

本稿では RePair を解析し，最頻出なペアを置換する RePair の動作が，最頻出な極大反復部分文字列の段階的な置換に対応することを示した．この解析に基づき，RePair を拡張した新たなアルゴリズム MR-RePair を開発した．MR-RePair は最頻出なペアの代わりに，最頻出な極大反復部分文字列に基づいて置換を行うものである．また，MR-RePair を実装し RePair と構築する文法サイズに関して比較実験を行った．その結果，特に繰り返しの多い文字列データに対して MR-RePair の有効性が確認できた．

今回，文法の符号化に関しては議論しなかったが，これは実用上非常に重要である．MR-RePair の構築した文法に対して，単純に区切り文字を挟んで各規則を並べて符号化する方法では，規則数が最終的なデータサイズに大きく影響する．MR-RePair のための効率よい符号化法の開発は今後の課題のひとつである．

謝辞 本研究は JSPS 科研費 JP17H06923, JP17H01697, JP16H02783，JP18H04098，JP18K11149 の助成を受けたものです．本研究はまた，JST，CREST，JPMJCR1402 の支援を受けたものです．本研究を遂行するにあたって，ソースコードを提供していただいたすべての方に感謝致します．

[*4] https://github.com/nicolaprezza/Re-Pair
[*5] https://github.com/rwanwork/Re-Pair; We ran it with level 0 (no heuristic option).
[*6] https://github.com/syoshid/Re-Pair-VF; We removed a routine to find the best rule set.
[*7] http://pizzachili.dcc.uchile.cl/repcorpus.html
[*8] http://corpus.canterbury.ac.nz/descriptions/\#large

参考文献

[1] Apostolico, A. and Lonardi, S.: Off-line compression by greedy textual substitution, *Proceedings of the IEEE*, Vol. 88, No. 11, pp. 1733–1744 (2000).
[2] Belazzougui, D. and Cunial, F.: Fast label extraction in the CDAWG, *International Symposium on String Processing and Information Retrieval*, Springer, pp. 161–175 (2017).
[3] Belazzougui, D., Cunial, F., Gagie, T., Prezza, N. and Raffinot, M.: Composite Repetition-Aware Data Structures, *Combinatorial Pattern Matching* (Cicalese, F., Porat, E. and Vaccaro, U., eds.), Cham, Springer International Publishing, pp. 26–39 (2015).
[4] Bille, P., Gørtz, I. L. and Prezza, N.: Space-Efficient Re-Pair Compression, *2017 Data Compression Conference (DCC)*, pp. 171–180 (2017).
[5] Charikar, M., Lehman, E., Liu, D., Ring, P., Prabhakaran, M., Sahai, A. and abhi shelat: The smallest grammar problem, *IEEE Transactions on Information Theory*, Vol. 51, No. 7, pp. 2554–2576 (2005).
[6] Claude, F. and Navarro, G.: Fast and compact web graph representations, *ACM Transactions on the Web (TWEB)*, Vol. 4, No. 4, p. 16 (2010).
[7] Gańczorz, M. and Jeż, A.: Improvements on Re-Pair Grammar Compressor, *Data Compression Conference (DCC), 2017*, IEEE, pp. 181–190 (2017).
[8] González, R. and Navarro, G.: Compressed text indexes with fast locate, *Annual Symposium on Combinatorial Pattern Matching*, Springer, pp. 216–227 (2007).
[9] Inenaga, S., Funamoto, T., Takeda, M. and Shinohara, A.: Linear-time off-line text compression by longest-first substitution, *International Symposium on String Processing and Information Retrieval*, Springer, pp. 137–152 (2003).
[10] Larsson, N. J. and Moffat, A.: Off-line dictionary-based compression, *Proceedings of the IEEE*, Vol. 88, No. 11, pp. 1722–1732 (2000).

表 1 実験に用いた文字列データ．

テキスト	サイズ (byte)	$\|\Sigma\|$	内容
rand77.txt	2,097,152	77	32 copies of 1024 random patterns of length 64
fib41	267,914,296	2	Fibonacci string from Pizza&Chili Corpus
einstein.de.txt	92,758,441	117	Edit history of Wikipedia for Albert Einstein
E.coli	4,638,690	4	Complete genome of the E. Coli bacterium
bible.txt	4,047,392	63	The King James version of the bible
world192.txt	2,473,400	94	The CIA world fact book

表 2 生成文法のサイズと実行時間の一覧．表の各マスの数字は，上から，生成された規則の数，開始記号からはじまる規則を除いたすべての規則の右辺の文字数の総和，開始記号からはじまる規則の右辺の文字数，全体の文法サイズである．水平線で区切られた 5 行目は実行時間を示し，単位は秒である．

	RePair					Re-PairImp	MR-RePair
text file	Maruyama	Navarro	Prezza	Wan	Yoshida		
rand77.txt	41,651	41,642	41,632	41,675	41,651	41,661	4,492
	83,302	83,284	83,264	83,350	83,302	83,322	46,143
	9	2	7	2	9	2	9
	83,311	83,286	83,271	83,352	83,311	83,324	**46,152**
	0.41	**0.37**	4.76	4.27	0.40	3.95	0.42
fib41	38	38	38	38	38	37	38
	76	76	76	76	76	74	76
	3	3	3	3	3	23	3
	79	**79**	**79**	**79**	**79**	97	**79**
	26.75	**23.94**	96.05	483.86	25.04	1360.40	33.62
einstein.de.txt	49,968	49,949	50,218	50,057	49,968	49,933	21,787
	99,936	99,898	100,436	100,114	99,936	99,866	71,709
	12,734	12,665	13,419	12,610	12,734	12,672	12,683
	112,670	112,563	113,855	112,724	112,670	112,538	**84,392**
	30.08	43.45	216.74	213.15	30.76	452.56	**29.63**
E.coli	66,664	66,757	66,660	67,368	66,664	66,739	62,363
	133,328	133,514	133,320	134,736	133,328	133,478	129,138
	651,875	649,660	650,538	652,664	651,875	650,209	650,174
	785,203	783,174	783,858	787,400	785,203	783,687	**779,312**
	1.20	**1.02**	14.67	10.37	1.56	27.04	1.33
bible.txt	81,193	81,169	80,999	81,229	81,193	81,282	72,082
	162,386	162,338	161,998	162,458	162,386	162,564	153,266
	386,514	386,381	386,992	386,094	386,514	385,989	386,516
	548,900	548,719	548,990	548,552	548,900	548,553	**539,782**
	1.33	**1.21**	13.00	9.12	1.47	24.38	1.27
world192.txt	55,552	55,798	55,409	55,473	55,552	55,437	48,601
	111,104	111,596	110,812	110,946	111,104	110,874	104,060
	213,131	213,962	213,245	212,647	213,131	212,857	212,940
	324,235	325,558	324,057	323,593	324,235	323,731	**317,000**
	0.59	0.80	7.57	4.89	**0.56**	12.35	0.66

[11] Nakamura, R., Bannai, H., Inenaga, S. and Takeda, M.: Simple Linear-Time Off-Line Text Compression by Longest-First Substitution, *Data Compression Conference, 2007. DCC '07*, pp. 123–132 (2007).

[12] Navarro, G. and Russo, L. M.: Re-pair Achieves High-Order Entropy., *DCC*, p. 537 (2008).

[13] Ochoa, C. and Navarro, G.: RePair and All Irreducible Grammars are Upper Bounded by High-Order Empirical Entropy, *IEEE Transactions on Information Theory*, pp. 1–5 (online), DOI: 10.1109/TIT.2018.2871452 (2018).

[14] Takagi, T., Goto, K., Fujishige, Y., Inenaga, S. and Arimura, H.: Linear-Size CDAWG: New Repetition-Aware Indexing and Grammar Compression, *String Processing and Information Retrieval* (Fici, G., Sciortino, M. and Venturini, R., eds.), Cham, Springer International Publishing, pp. 304–316 (2017).

[15] Wan, R.: Browsing and searching compressed documents, PhD Thesis (2003).