SATソルバを用いたNP困難な圧縮指標の高速計算

Fast Computation of NP-hard Repetitiveness Measures using SAT solvers

坂内 英夫[1]　後藤 啓介[2]*　石畠 正和[3]
神田 峻介[2]　クップル ドミニク[1]　西本崇晃[4]
Hideo Bannai[1]　Keisuke Goto[2]　Masakazu Ishihata[3]
Shunsuke Kanda[2]　Dominik Köppl[1]　Takaaki Nishimoto[4]

[1] 東京医科歯科大学 (Tokyo Medical and Dental University)
[2] 無所属 (Independent)　[3] 日本電信電話株式会社 (NTT Corp.)　[4] 理化学研究所 (RIKEN)

Abstract: Repetitiveness measures reveal profound characteristics of datasets, and give rise to compressed data structures and algorithms working in compressed space. Alas, the computation of some of these measures is NP-hard, and infeasible for datasets of even small sizes. Two such measures are the smallest size γ of a string attractor, and the smallest size b of a bidirectional macro scheme. For these two measures, we study their efficient computation using SAT solvers returning instances attaining these smallest sizes.

1　はじめに

インターネットの普及やセンサ機器の発達により取得可能なデータは年々増えている。天文学の分野では1時間あたり数テラバイトのデータを生成する望遠鏡の開発が行われ、またウェブでは既に600億ものページ、4ペタバイトのテキストコンテンツが存在しかつまだまだ増え続けているとされている [9]。この様な大規模データをそのままの状態で保存したり処理することは現実的ではなく、何らかの形式で圧縮して保存し、また圧縮したまま処理する事が必要である。

データがどの程度圧縮できるかはデータに含まれる情報の量あるいは冗長性を表すため、その見積もりは重要である。代表的な指標として、シャノンの統計エントロピーやそれを拡張した経験エントロピーが知られているが、これらの指標は反復の多いデータに対しては冗長な量を正確に捉えることが出来ない。またコロモゴロフ複雑性 [7] は冗長な量を正確に捉えることができるものの、計算不能であることが知られている。近年は、反復の多いデータに対して冗長な量を捉える良い圧縮指標として辞書式圧縮基づく圧縮表現のサイズが用いられることが多く、それら各種圧縮指標の関係性の理論的解析が盛んに行われている [9]。理論解析の手助けとなるのが計算機による圧縮指標の厳密計算である。しかしながら、いくつかの圧縮表現については最小のものを求めるのはNP困難であり、計算機による厳密計算は小さいサイズのデータに限られているのが現状である。さらなる解析の発展のためにはこれらNP困難な圧縮指標の高速計算方法の開発が必要である。

本論文ではNP困難な圧縮指標: 最小文字列アトラクタ [6] と最小Bidirectional Macro Schemes [12] のSAT定式化を行い、高速なSATソルバを用いたNP困難な圧縮指標の高速な計算を実現する。

2　準備

2.1　文字列

文字列$T[1..n]$はアルファベットΣから得られる長さnの文字の並びである。長さnの文字列T、位置$1 \leq i \leq j \leq n$について、$T[i..j]$をiから始まりjで終わるTの部分文字列、$occ(T[i..j]) = \{i' \mid T[i'..i'+(j-i)] = T[i..j]\}$を

*E-mail：keisukegotou@gmail.com

$T[i..j]$ の T 中の出現位置集合、$cover(T[i..j]) = \{k \mid \exists i' \in occ(s), i' \leq k \leq i' + j - i\}$ を T 中における $T[i..j]$ のカバー位置集合とする。また、$k \in cover(T[i..j])$ のとき、位置 k は部分文字列 $T[i..j]$ をカバーすると言う。

2.2 文字列アトラクタ

長さ n の文字列 T について、T の k-文字列アトラクタ [6] とは次の性質を満たす位置集合 $\Gamma \subseteq [1..n]$ である。T の任意の長さ k 以下の部分文字列 $T[i..j]$（$i \leq j < i + k$）について、$p \in cover(T[i..j])$ なる位置 $p \in \Gamma$ が存在する。また、$k = n$ のとき単純に Γ を T の文字列アトラクタと呼ぶ。

2.3 Bidirectional Macro Schemes

Bidirectional Macro Schemes (BMS) [12] とは文字列 T をフレーズ $T = f_1 \dots f_m$ に分解し、各フレーズ f_i を単一の文字 $c \in \Sigma$、あるいは部分文字列 $T[b..e]$ の参照を示す開始位置と終了位置のペア (b, e) で表す文字列表現である (ただし $1 \leq b \leq e \leq n$)。BMS は参照がループする場合、表す文字列が一意に定まらない。BMS は単一の文字列を生成するとき正当（valid）であると呼ぶ。

2.4 SAT 問題

充足可能性判定問題（SAT 問題） [4] とは与えられた命題論理式を真にする値割当が存在するかを判定する問題である。SAT 問題は通常、いくつかの節の連言（AND）、各節がリテラル（ブール変数あるいはその否定）の選言（OR）の形式を取る和積標準形（CNF）で表される事が多い。CNF 式で表されるとき、すべての節を充足する値割当を求めることが目的となる。MAX-SAT は SAT の拡張問題であり、充足性に対しての条件が緩和されている。MAX-SAT の CNF 式では hard 節と soft 節の 2 種類が存在し、hard 節は必ず充足する必要があるが、soft 節は必ず充足する必要はない。soft 節には重みが設定されており、MAX-SAT はすべての hard 節を充足し、充足する soft 節の重みの総和を最大化する値割当を求める問題である。

疑似ブール制約はブール変数上の線形制約であり、以下の形式で表される。リテラル列 $X = (x_1, \dots, x_m)$、整数リテラル a_i、c、関係演算子 $\triangleright \in \{\leq, \geq, =\}$ について、$\sum_{i \in [1,m]} a_i x_i \triangleright c$。

その他にも条件制約 $x \to y$、$x \leftrightarrow y$ などがあり、これらの制約や疑似ブール制約は補助変数を導入することによりすべて CNF に変換できることが知られている。本論文では上記の制約を用いて各種圧縮指標の定式化を行い、その後に CNF 式への変換を行う。

3 最小文字列アトラクタの制約

まず、3.1節において、文字列アトラクタの定義から愚直に導かれる制約を考える。極小部分文字列の性質に着目すると愚直な制約は冗長であり、より単純化することができることが分かる。ソルバの高速化のためにも冗長な制約を取り除いたほうが良い。3.2節では極小部分文字列の性質を用いて愚直な制約から冗長な制約を排除した、より単純化した制約を考える。

3.1 愚直な制約

文字列 T の各位置 $i \in [1, n]$ に対して、リテラル p_i を作成し、$p_i = 1$ であるとき、かつそのときに限り位置 i はアトラクタの要素とする。T に出現するすべての部分文字列集合を $S = \{T[i..j] \mid 1 \leq i, j \leq n\}$ とする。この任意の要素 $s_k \in S$ について、hard 節 $C_k = \bigvee_{i \in cover(s_k)} p_i$ を作成する。また、soft 節 $D_i = \neg p_i$ を作成し、重みを 1 に設定する。hard 節は文字列アトラクタの定義を愚直に制約化したものであるため、MAX-SAT は文字列アトラクタの定義を満たすもののうち最小サイズのアトラクタを求める。

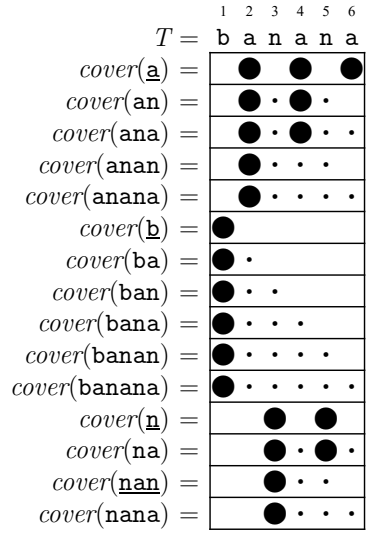

図 1: 文字列 $T = \texttt{banana}$ の各部分文字列の出現位置集合とカバー位置集合。$cover(x)$ の行にある位置 k の点はその位置 k が T の部分文字列 x をカバーしていることを意味する。さらに、その位置が x の出現位置でもある場合には、大きな点で表している。また、下線を引かれた部分文字列は T の極小部分文字列である。

3.2 極小部分文字列による制約の単純化

極小部分文字列の性質を利用して hard 節数の削減を行う。

T の任意の部分文字列 x, y について、(i) $cover(x) \subseteq cover(y)$ もしくは $cover(y) \subseteq cover(x)$ を満たし、かつ (ii) $|occ(x)| = |occ(y)|$ を満たすとき、x と y を同値とするような同値関係を考える。このとき、x と同値であるような x の真の部分文字列（すなわち、x より短い x の部分文字列）が存在しないとき、x を T の極小部分文字列と呼ぶ。図 1は文字列 $T = \texttt{banana}$ の各部分文字列の出現位置集合とカバー位置集合を表している。この例では、部分文字列 nan の出現位置集合とカバー位置集合はそれぞれ $occ(\texttt{nan}) = \{3\}$ と $cover(\texttt{nan}) = \{3, 4, 5\}$ である。同様に、部分文字列 nana の出現位置集合とカバー位置集合はそれぞれ $occ(\texttt{nana}) = \{3\}$ と $cover(\texttt{nana}) = \{3, 4, 5, 6\}$ である。$cover(\texttt{nan}) \subseteq cover(\texttt{nana})$ かつ $|occ(\texttt{nan})| = |occ(\texttt{nana})|$ なので、２つの部分文字列 nan と nana は同値である。さらに、nan と同値であるような nan の真の部分文字列は存在しないので、nan は極小部分文字列である。

任意の部分文字列の任意の出現はそれと同値な極小部分文字列の出現を部分文字列として含むので、極小部分文字列をすべてカバーする位置集合は T の文字列アトラクタとなる。一方で、T の任意の文字列アトラクタは極小部分文字列をすべてカバーしているので、文字列アトラクタは極小部分文字列をすべてカバーする位置集合を表している。したがって、与えられた位置集合が T の文字列アトラクタであるとき、かつそのときに限り、その集合は T の全ての極小部分文字列をすべてカバーする。

3.1節の愚直な制約によって得られた CNF 式から、極小部分文字列ではない部分文字列を用いて作成した hard 節を全て取り除くことによって、hard 節数を削減する。このとき、残った hard 節は各極小部分文字列を用いて作成した hard 節と一致する。そして、上述の極小部分文字列をすべてカバーする位置集合の性質から、与えられた値割当が元々の hard 節をすべて充足するとき、かつそのときに限り、その値割当は削減後の hard 節をすべて充足するという事実が導かれる。したがって、愚直な制約を用いた MAX-SAT 問題と同様に、hard 節を削減したあとの MAX-SAT 問題を解くことによっても T の最小の文字列アトラクタを求めることができる。

4 最小 BMS の制約

正当な BMS の制約化を行う。各フレーズ長が高々k の BMS を k-BMS と呼び、n-BMS を単に BMS と呼ぶ。任意の BMS は参照位置を維持したままフレーズ長を 1 に制限した 1-BMS へと愚直に変換することができる。正当な 1-BMS は後述するように背後に木構造を持ち、それらの構造は相互に変換可能である。提案手法ではまず木

構造の制約化を行い、その木構造に対応する 1-BMS の制約化、また 1-BMS から対応するフレーズ数最小の BMS の制約化を行うことで最小 BMS を求める。

1-BMS において各位置をノードとみなすと、正当な 1-BMS はループを持たないノードの連結で表され、その参照構造は全体で木集合を表す。$M_i = \{j \in [1,n] \mid T[i] = T[j], i \neq j\}$ を $T[i]$ と同じ文字が出現する i 以外の位置集合とする。ここで深さ $d \in [1,n]$、各位置 $i \in [1,n], j \in M_i$ について、$dref_{d,i,j} = 1$ であるとき、かつそのときに限り、深さ d の位置 i が深さ $d-1$ の位置 j を参照しているとする（位置 i の親が位置 j）。また、位置 i について $root_i = 1$ であるとき、かつそのときに限り位置 i が木構造の根であるとする。

ここで木構造の連結性から以下の制約が導かれる。

$$\forall i \in [1,n], root_i + \sum_{d \in [1,n], j \in M_i} dref_{d,i,j} = 1 \tag{1}$$

$$\begin{aligned} &\forall d \in [2,n], \forall i \in [1,n], \forall j \in M_i, \\ &dref_{d,i,j} \rightarrow \sum_{k \in M_j} dref_{d-1,j,k} = 1 \end{aligned} \tag{2}$$

木構造に対応する 1-BMS の制約を考える。位置 i と位置 $j \in M_i$ について $ref_{i,j} = 1$ であるときに限り、i が j を参照するとする。

各位置について参照の数はたかだか 1 である。

$$\forall i \in [1,n], \sum_{j \in M_i} ref_{i,j} \leq 1 \tag{3}$$

木構造の制約から以下の参照が求まる。

$$\begin{aligned} &\forall d \in [1,n], \forall i \in [1,n], \forall j \in M_i, \\ &dref_{d,i,j} \rightarrow ref_{i,j} \end{aligned} \tag{4}$$

$$\begin{aligned} &\forall i \in [1,n], \forall j \in M_i, \\ &root_i \rightarrow \neg ref_{i,j} \end{aligned} \tag{5}$$

1-BMS と同じ参照構造を持つフレーズ数最小の BMS の制約を考える。位置 i について $fbeg_i = 1$ であるときかつそのときに限り、i はフレーズの開始位置とする。つまり $fbeg_i = 1$ からそれに続く $fbeg_j = 0$ の連続位置が BMS のフレーズに対応する。

木構造の根は長さ 1 のフレーズに必ずなる。

$$root_i \rightarrow fbeg_i \tag{6}$$

参照元 i と参照先 j のそれぞれ 1 つ前の位置について、いずれかが 1 未満の場合。i がフレーズの途中になることはありえないため i はフレーズの開始位置となる。また同様のことが $T[i-1] \neq T[j-1]$ の場合にも言える。

$$\begin{aligned} &\forall i \in [1,n], \forall j \in M_i, \\ &ref_{i,j} \wedge (i = 1 \vee j = 1 \vee T[i-1] \neq T[j-1]) \rightarrow fbeg_i \end{aligned} \tag{7}$$

位置 i と $i-1$ の参照先が連続しないならば i と $i-1$ を同じフレーズに含む事が出来ないので i はフレーズの開始位置とする。

$$\begin{aligned} &\forall i \in [2,n], \forall j \in M_i \text{ s.t. } j > 1, \\ &\neg ref_{i-1,j-1} \wedge ref_{i,j} \rightarrow fbeg_i \end{aligned} \tag{8}$$

位置 i と $i-1$ の参照先が連続する場合、i と $i-1$ を同じフレーズに含めた場合のフレーズ数が、同じフレーズに含めなかった場合のフレーズ数以下に必ずなる。そのため、i をフレーズの開始位置としない。

$$\begin{aligned} &\forall i \in [2,n], \forall j \in M_i \text{ s.t. } (j-1) \in M_{i-1}, \\ &ref_{i-1,j-1} \wedge ref_{i,j} \rightarrow \neg fbeg_i \end{aligned} \tag{9}$$

上記制約を hard 節とし、soft 節として $\neg fbeg_i$ を作成し、重みを 1 に設定する。以上の定式化により、正当な BMS のうち最小の割当が求まる。

file	n	lz77	lzrr	lex	lcp	SA	BMS
bib	111261	15343	14091	15216	15769	10371	M
book1	768771	110043	100912	109029	115168	T	M
book2	610856	75430	69620	74563	77299	T	M
geo	102400	38246	35707	37837	39679	21590	M
news	377109	56462	52150	55901	58233	T	M
obj1	21504	7032	6854	6960	7081	3866	M
obj2	246814	41582	39138	40961	41985	T	M
paper1	53161	9261	8561	9158	9411	6355	M
paper2	82199	13805	12675	13495	13949	9884	M
paper3	46526	9063	8336	8889	9216	6295	M
paper4	13286	3273	3016	3192	3271	2055	M
paper5	11954	3051	2833	3005	3062	1879	M
paper6	38105	7079	6560	6970	7139	4668	M
pic	513216	25418	23531	25253	26714	T	M
progc	39611	7144	6634	6993	7153	4714	M
progl	71646	7993	7437	7854	8048	T	M
progp	49379	5751	5398	5660	5790	T	M
trans	93695	9089	8401	8881	9135	T	M
fib03	2	2	2	2	2	2	2
fib04	3	3	3	3	3	2	3
fib05	5	4	4	4	4	2	4
fib06	8	5	4	4	4	2	4
fib07	13	6	5	5	5	2	4
fib08	21	7	5	4	4	2	4
fib09	34	8	5	6	6	2	4
pds01	2	2	2	2	2	2	2
pds02	4	4	4	4	4	2	4
pds03	8	6	5	5	5	2	5
pds04	16	8	6	7	7	2	6
pds05	32	10	7	7	7	2	7
pds06	64	12	8	10	10	2	7
pds07	128	14	9	9	9	2	M
thuemorse01	2	2	2	2	2	2	2
thuemorse02	4	4	4	4	4	2	4
thuemorse03	8	6	6	6	6	3	5
thuemorse04	16	8	7	7	7	4	6
thuemorse05	32	10	8	9	9	4	7
thuemorse06	64	12	9	10	10	4	8
thuemorse07	128	14	10	12	12	4	M

表 1: 出力サイズの比較。M、T はそれぞれメモリー不足、3600 秒のタイムアウトにより計算が完了しなかったことを意味する。

5 計算機実験

最小文字列アトラクタ（SA）と最小 BMS（BMS）の制約を実装、文字列コーパスと典型的な文字列系列に対して適用し、その時の計算時間と出力サイズを測定した。参考としてその他の圧縮法として多項式時間で計算可能な圧縮指標 LZ77 [8]、LZRR [11]、lex-parse [10]、lcp-comp [5] と比較した。本来であれば、ナイーブな計算手法と比較し、どれだけ計算時間の向上が見られたか比較すべきであるが、今回は時間の都合上行っていない。データセットには、The Canterbury Corpus[1]で提供される The Calgary Corpus（テキストファイル 14 種類）、文字列系列としてフィボナッチ文字列、Period-doubling 文字列 [3]、Thue-Morse 文字列 [2] を使用した。

計算機は Windows 10 Pro、Intel Core i7-9700 3.00GHz 32GB メモリを使用し、各プログラムはシングルスレッドで実行した。SAT ソルバは PySAT ライブラリを使用した [2]。既存の圧縮指標の計算は lzrr [3]の実装を使用した。

表 1は各圧縮指標の比較を行っている。最小 BMS は LZ77 (lz77)、LZRR (lzrr)、lex-parse (lex)、lcp-comp (lcp) の下限となることが理論的に示されており、実験結果からもその点が確認できる。また最小文字列アトラクタは最小 BMS の下限であることも理論的に示されており、その点も実験結果から確認できる。

表 2は各圧縮指標の計算にかかる時間の比較を行っている。文字列アトラクタは既存の多項式時間の圧縮指標よりも計算時間がかかり、最小 BMS は最小文字列アトラクタよりもさらに計算時間がかかる傾向にある。このことは最小 BMS の定式化が最小文字列アトラクタの定式化よりも複雑であることが大きな要因であると考えられる。

[1] https://corpus.canterbury.ac.nz/
[2] https://pysathq.github.io/
[3] https://github.com/TNishimoto/lzrr

file	n	lz77	lzrr	lex	lcp	SA	BMS
bib	111261	0.0769	0.0774	0.0610	2.8035	327.3213	M
book1	768771	0.1259	0.2673	0.1231	166.4935	T	M
book2	610856	0.0933	0.1868	0.0768	84.0316	T	M
geo	102400	0.0564	0.0722	0.0410	6.8832	628.0244	M
news	377109	0.0704	0.1045	0.0556	35.8974	T	M
obj1	21504	0.0359	0.0317	0.0320	0.2323	50.8647	M
obj2	246814	0.0510	0.0697	0.0475	14.4359	T	M
paper1	53161	0.0360	0.0339	0.0335	0.8442	187.3785	M
paper2	82199	0.0519	0.0434	0.0356	1.9029	229.6118	M
paper3	46526	0.0407	0.0397	0.0367	0.7464	82.8846	M
paper4	13286	0.0363	0.0321	0.0472	0.0991	7.2535	M
paper5	11954	0.0292	0.0297	0.0288	0.0896	6.3755	M
paper6	38105	0.0356	0.0327	0.0301	0.4629	47.9410	M
pic	513216	0.0540	0.1096	0.0579	12.3684	T	M
progc	39611	0.0393	0.0413	0.0366	0.4915	50.3092	M
progl	71646	0.0417	0.0410	0.0383	0.8519	T	M
progp	49379	0.0312	0.0324	0.0318	0.4309	T	M
trans	93695	0.0411	0.0444	0.0393	1.2154	T	M
fib03	2	0.0276	0.0288	0.0372	0.0246	0.1437	0.0010
fib04	3	0.0344	0.0301	0.0279	0.0235	0.1329	0.0014
fib05	5	0.0347	0.0269	0.0283	0.0245	0.1380	0.0025
fib06	8	0.0278	0.0334	0.0226	0.0227	0.1375	0.0101
fib07	13	0.0307	0.0274	0.0272	0.0262	0.1414	0.0224
fib08	21	0.0287	0.0265	0.0249	0.0233	0.1428	0.0789
fib09	34	0.0268	0.0324	0.0271	0.0246	0.1362	0.3704
pds01	2	0.0235	0.0237	0.0220	0.0229	0.1344	0.0011
pds02	4	0.0273	0.0253	0.0223	0.0265	0.1390	0.0022
pds03	8	0.0333	0.0286	0.0291	0.0352	0.1432	0.0061
pds04	16	0.0297	0.0306	0.0315	0.0295	0.1462	0.0482
pds05	32	0.0318	0.0291	0.0382	0.0330	0.1435	2.3336
pds06	64	0.0374	0.0303	0.0322	0.0301	0.1404	50.1594
thuemorse01	2	0.0358	0.0442	0.0386	0.0321	0.1353	0.0012
thuemorse02	4	0.0390	0.0297	0.0287	0.0278	0.1595	0.0017
thuemorse03	8	0.0288	0.0275	0.0285	0.0322	0.1918	0.0048
thuemorse04	16	0.0277	0.0321	0.0327	0.0352	0.1495	0.0290
thuemorse05	32	0.0380	0.0293	0.0335	0.0300	0.1420	0.2764
thuemorse06	64	0.0327	0.0302	0.0289	0.0313	0.1394	25.4202
thuemorse07	128	0.0287	0.0310	0.0417	0.0299	0.1627	M

表 2: 計算時間の比較。単位は秒を表し、M、T はそれぞれメモリー不足、3600 秒のタイムアウトにより計算が完了しなかったことを意味する。

6 ソルバを用いた γ の感度解析

Akagi ら [1] は圧縮手法の感度の概念を提案した。ある圧縮手法 C による長さ n の文字列 T の圧縮サイズを $C(T)$ としたとき、C の乗算感度 $MS(C,n)$ は以下のように定義される

$$MS_{op}(C,n) = \max_{T\in\Sigma^n, T'\in\Sigma^*} \left\{ \frac{C(T')}{C(T)} \;\middle|\; ed_{op}(T,T') = 1 \right\}。$$

ここで、$ed_{op}(T,T')$ は T に対する 1 回の編集操作 op により T' が得られることを意味し、op として は 1 文字の置換・挿入・削除などの操作が考えられる。Akagi らは様々な圧縮手法に対して 乗算感度のタイトな上界と下界を示しているが文字列アトラクタの乗算感度については 1 文字の置換・挿入・削除いずれに対しても $O(\log n)$ の上界と 2 の下界しか得られておらず大きなギャップがある。そこで、提案手法を用いて短い長さのバイナリ文字列に対して全数探索を行ったところ、任意の $k \geq 2$ に対して $\gamma(\mathtt{abbb}\underline{\mathtt{a}}\mathtt{aa}\underline{\mathtt{b}}^k) = 2$ ($k = 2$ のとき $\{4,7\}$ が、 $k > 2$ のとき $\{5,8\}$ が最小文字列アトラクタとなる), 任意の $k' \geq 4$ に対して $\gamma(\underline{\mathtt{a}}\mathtt{bb}\underline{\mathtt{b}}\mathtt{a}\underline{\mathtt{a}}\mathtt{ab}\underline{\mathtt{cb}}^{k'}) = 5$ ($\{1,4,6,9,10\}$ は文字列アトラクタである。一方で $\mathtt{abb}, \mathtt{ba}, \mathtt{aab}, \mathtt{c}, \mathtt{b}^{k'}$ の 5 つの部分文字列はいずれも 1 度しか出現せず、それぞれをカバーする位置がアトラクタに含まれる必要があるため、最小の文字列アトラクタである) であることを発見した。これは、任意の $n \geq 13$ と置換・挿入操作に対しても $MS_{op}(\gamma,n) \geq 2.5$ であることを示しており、Akagi らの示した乗算感度の下界を改善している。

7　おわりに

本論文では NP 困難な圧縮指標である文字列アトラクタと Bidirectional Macro Schemes を SAT ソルバを用いて計算する手法を提案した。今後の課題として、より詳細な比較実験の実施と、それに加えその他の NP 困難な圧縮指標、例えば最小文脈自由文法の制約化が考えられる。

参考文献

[1] Tooru Akagi, Mitsuru Funakoshi, and Shunsuke Inenaga. Sensitivity of string compressors and repetitiveness measures. *CoRR*, abs/2107.08615, 2021.

[2] Jean-Paul Allouche and Jeffrey Shallit. The ubiquitous prouhet-thue-morse sequence. In *Sequences and their applications*, pages 1–16. Springer, 1999.

[3] Jean-Paul Allouche, Jeffrey Shallit, et al. *Automatic sequences: theory, applications, generalizations*. Cambridge university press, 2003.

[4] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2009.

[5] Patrick Dinklage, Johannes Fischer, Dominik Köppl, Marvin Löbel, and Kunihiko Sadakane. Compression with the tudocomp framework. In *Proc. SEA*, volume 75 of *LIPIcs*, pages 13:1–13:22, 2017.

[6] Dominik Kempa and Nicola Prezza. At the roots of dictionary compression: string attractors. In *Proc. STOC*, pages 827–840. ACM, 2018.

[7] Andrei N Kolmogorov. Three approaches to the quantitative definition of information. *Problems of information transmission*, 1(1):1–7, 1965.

[8] Abraham Lempel and Jacob Ziv. On the complexity of finite sequences. *IEEE Transactions on information theory*, 22(1):75–81, 1976.

[9] Gonzalo Navarro. Indexing highly repetitive string collections, part I: Repetitiveness measures. *ACM Computing Surveys (CSUR)*, 54(2):1–31, 2021.

[10] Gonzalo Navarro, Carlos Ochoa, and Nicola Prezza. On the approximation ratio of ordered parsings. *IEEE Transactions on Information Theory*, 67(2):1008–1026, 2020.

[11] Takaaki Nishimoto and Yasuo Tabei. LZRR: LZ77 parsing with right reference. *Information and Computation*, page 104859, 2021.

[12] James A Storer and Thomas G Szymanski. Data compression via textual substitution. *Journal of the ACM (JACM)*, 29(4):928–951, 1982.