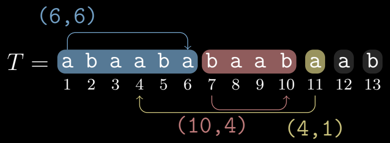
Answer Set Programming を用いた圧縮指標の計算

クップル ドミニク[1]　番原 睦則[2]

[1]: 山梨大学, [2]: 名古屋大学



BMS: (6,6)(10,4)(4,1)ab

LA Symposium 2023 [28]

圧縮指標

- 反復の多いデータが大量に増加している
- 膨大なデータを保存や処理をするため、効果的な圧縮を求めたい

質問

データをどのぐらい圧縮できるか？→ 圧縮指標で測る

圧縮指標は

- 圧縮方法の性格を定める
- 下界を定める（どのぐらい圧縮できるか）

圧縮指標の一覧

指標	時間量	解決方法
LZ (LZ77, LZ78)	$\mathcal{O}(n)$	
BW 変換の連	$\mathcal{O}(n)$	
アトラクタ	NP 困難	SAT [Bannai+'22]
最小の BMS	NP 困難	SAT [Bannai+'22]
最小の SLP	NP 困難	SAT [Bannai+'22]
最小の RLSLP	NP 困難	SAT [Kawamoto+'24]

- BMS: Bidirectional Macro Scheme
- SLP: straight-line program (文脈自由文法の一つ)
- RLSLP: 連超圧縮できる SLP

今回の発表で最小の BMS に注目

ファイル		計算時間 (秒)	
名前	n	SAT	ASP
BIB	128	0.28	0.02
BIB	256	44.13	0.49
BOOK1	128	0.22	0.02
BOOK1	256	N/A	0.13
BOOK2	128	0.55	0.02
BOOK2	256	N/A	0.14
NEWS	128	0.15	0.02
NEWS	256	13.61	0.27
OBJ2	128	3.64	0.04
OBJ2	256	N/A	19.89
PAPER1	128	0.56	0.09
PAPER1	256	25.18	0.12
PAPER2	128	0.26	0.27
PAPER2	256	29.78	27.29
PAPER4	128	0.19	0.04
PAPER4	256	31.75	0.46
PROGC	128	0.44	0.02
PROGC	256	N/A	0.21
PROGL	128	18.23	0.15
PROGL	256	N/A	39.84
PROGP	128	0.24	0.02
PROGP	256	N/A	1.10
TRANS	128	0.45	0.03
TRANS	256	N/A	2.65

実験

- Calgary corpus の各ファイルに対して、最小の BMS の計算
- n はファイルから読んだ接頭辞の長さを示す
- 既存研究： SAT
- 提案する方法：ASP
- N/A： 1 分の計算時間制限を超えたことを示す
- SAT は 128 文字に対して計算できるが、256 文字から厳しくなる

本研究の貢献

- 最小の BMS を ASP 言語で高速に計算する手法を提案

設定

入力

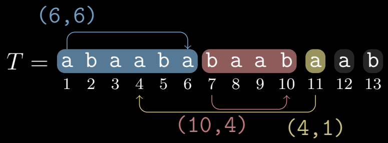
- $T[1..n]$: 文字列
- $T[i..j]$: i から始まり j で終わる T の部分文字列
- $T[i]$: T の i 番目の文字
- $T[i] \in \Sigma$, Σ: アルファベット
- n: T の長さ

定義 (分解)

文字列分解 とは T を複数の部分文字列 $F_1, \cdots, F_z$ に分解する

- ただし， $T = F_1 \cdots F_z$ である
- 各部分文字列 F_x を項と呼ぶ
- 項 F_x の開始位置と F_x の長さは、
- dst_x と λ_x がそれぞれ示している
- 言い換えると， $F_x = T[\mathsf{dst}_x..\mathsf{dst}_x + \lambda_x)$

目的： 長い項を $\mathcal{O}(1)$ 領域で表現すれば、圧縮率を求めることができる



BMS: (6,6)(10,4)(4,1)ab

BMS, Storer'82

定義 (BMS)

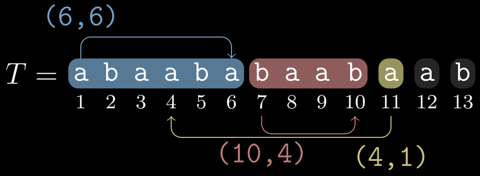
$T[1..n] = F_1 \cdots F_z$ を $F_1, \ldots, F_z$ の項に分解する BMS は，以下の状況を満たす

1. 各 $x \in [1..z]$ に対して $|F_x| \geq 2$ の場合，F_x は参照先 $\mathrm{src}_x \in [1..n]$ を持つ ただし，$T[\mathrm{src}_x..\mathrm{src}_x + \lambda_x) = F_x$ である
2. 説明のため，関数 $R(\mathrm{dst}_x) := \mathrm{src}_x \forall x$ を定義する (R は参照元から参照先に写像する) R の定義域を広げた場合，

$$\begin{aligned} &\forall x \in [1..z], |F_x| \geq 2 : \\ R(\mathrm{dst}_x + k) = \mathrm{src}_x + k \ &\forall k \in [0..|F_x|) \end{aligned} \tag{1}$$

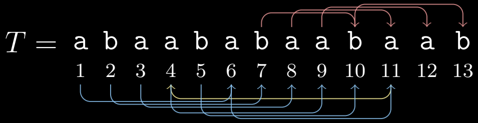
となり，グラフ (V, E) は閉路を持ち得ないただし，V はテキスト位置の集合 $[1..n]$ と $E := \{(i, R(i)) \mid i \in V, \text{定義された } R(i)\}$ である

貪欲法は最適？



BMS: (6,6)(10,4)(4,1)ab

- 左から文字を読む貪欲法
- 項の個数： 5
- $F_1 = T[1..6]$, $\mathrm{dst}_1 = 1$, $\lambda_x = 6$, $\mathrm{src}_1 = 6$



グラフ

- $V = [1..n]$,
- $E = \{(i, R(i)) \mid$ 定義された $R(i)\}$

に対して，最長の経路は
$3 \to 8 \to 11 \to 4 \to 9 \to 12$ である
観察結果：閉路の有無の確認は簡単ではないと言える

貪欲法は最適ではない

(6,6)

$T =$

a	b	a	a	b	a	b	a	a	b	a	a	b
1	2	3	4	5	6	7	8	9	10	11	12	13

(1,5)

BMS: (6,6)ba(1,5)

- 項の個数：4

(9,5)

$T =$

a	b	a	a	b	a	b	a	a	b	a	a	b
1	2	3	4	5	6	7	8	9	10	11	12	13

(1,6)

BMS: (9,5)(1,6)ab

- 項の個数：4

複数の最小の BMS が存在しうるが、貪欲法は最小にならない可能性がある

R から BMS, Bannai+'22

- 右図の上で、R が定義された
- R から項を帰着できる
- 定義したグラフ (V, E) は森構造になると確認すべき

$$T = \mathtt{a\ b\ a\ a\ b\ a\ b\ a\ a\ b\ a\ a\ b}$$

1 2 3 4 5 6 7 8 9 10 11 12 13

森構造から，以下のことを定める

- $V = [1..n]$,
- $E = \{(i, R(i)) \mid 定義された\ R(i)\}$

- 参照先を持たない位置 v は長さ 1 の項になる
- v は参照先 w があるとすれば，w は v の親である
- 2 つの隣接なテキスト位置 $T[i]$ と $T[i+1]$ は下記の 2 つの条件を満たせば，同じ項に含める可能性がある
  1. i と $i+1$ は親が持つ，つまり $R(i), R(i+1) \in [1..n]$ は定義されている
  2. $R(i+1) = R(i) + 1$ である（図のようにずらしている参照先）

つまり，項の開始位置 p_i と R の選択で BMS を帰着できる

充足可能性問題（SAT 問題）

定義 (SAT 問題)

- 日本語：充足可能性問題
- 入力：和積標準形（CNF）形式で表される論理式、すなわち節の集合

定義 (節)

- 連言 ($\wedge$) の連続されているブール変数を節と呼ぶ
- 節の例：$x_1 \wedge \neg x_2 \wedge \cdots \wedge x_j$

CNF において、SAT はすべての節を充足する（真にする）割当を求める

ASP 言語, Gebser+'12

解集合プログラミング (ASP; Answer Set Programming)

- SAT の拡張
- 簡単に言えば、整数の変数を節の中に使用可能

ASP による問題解法

$$\begin{tikzcd}
\text{問題} \arrow[r, "\text{1. モデリング}"] & \text{論理プログラム} \arrow[r, "\text{2. ASP システム}"] & \text{解集合} \arrow[d, "\text{3. 解釈}"'] \\
& & \text{問題の解}
\end{tikzcd}$$

1. 解きたい問題を論理プログラムとしてモデリングする
2. ASP システムを用いて，論理プログラムの解集合 (一種の最小モデル) を計算する
3. 解集合を解釈して元の問題の解を得る

論理プログラムとルール

ASP 言語は論理プログラムをベースとしている

- 論理プログラム P とは，以下の形式のルール r の有限集合である

ASP	論理プログラム
$\underbrace{a_0}_{\text{ヘッド}}$:- $\underbrace{a_1, a_2, \ldots a_n}_{\text{ボディ}}$.	$a_1 \wedge a_2 \wedge \ldots \wedge a_n \implies a_0$

- ただし、各 a_i はアトム、‘,’ は連言 ($\wedge$) を表す
- 直観的な意味は、すべて $i \in [1..n]$ に対して、 a_i は成り立つ場合、a_0 が成り立つ

ASP: ファクトと節

- ボディが空のルールをファクトと呼び，:- を省略可能

$$\underbrace{a_0}_{\text{ヘッド}}.$$

- ヘッドが空のルールを節と呼ぶ (SAT のように)

ASP

$$:\!\text{-}\ \underbrace{a_1, a_2, \ldots a_n}_{\text{ボディ}}.$$

ボディが成り立たないことを表す

論理プログラム

$$\neg(a_1 \wedge a_2 \wedge \ldots a_n)$$

拡張構文

ASP 言語には，組合せ問題を解くために便利な構文が用意されている

- 選択肢

ASP	論理プログラム
$\{a_1; a_2; \ldots; a_n\}$	$a_1 \vee a_2 \vee \ldots \vee a_n$

  アトム集合 $\{a_1, a_2, \ldots, a_n\}$ の任意の部分集合が成り立つことを意味する

- 個数制約

ASP	論理プログラム
$\ell\,\{a_1; a_2; \ldots; a_n\}\,m$	$\lvert\{i \in [1..n] : a_i$ は真である $\}\rvert \in [\ell..m]$

  - $a_1, a_2, \ldots, a_n$ のうち，ℓ 個以上 m 個以下が成り立つことを意味する
  - ℓ と m を書かないと、 $\ell = 0$ と $m = n$ とみなす

ASP $\leftrightarrow$ BMS 翻訳テーブル

演算	論理プログラム	ASP
a の否定	$\neg a$	not a.
a の上で b	$a \implies b$	b :- a.
a は真		a.

これら以外にも，組合せ最適化問題を解くための最小化関数・最大化関数なども用意されている

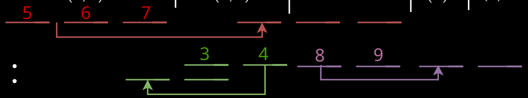
BMS
最小の Bidirectional Macro Scheme

BMS Input

Σ を整数に写像 (例：ASCII)

- a ↦ 97, b ↦ 98

$$T = \begin{array}{ccc|cc|cc|c|c} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ \texttt{a} & \texttt{b} & \texttt{a} & \texttt{a} & \texttt{a} & \texttt{b} & \texttt{a} & \texttt{b} & \texttt{a} \\ & (5,7) & & \multicolumn{2}{c|}{(3,4)} & & & (\mathrm{b}) & (\mathrm{a}) \end{array}$$



方針：

- BMS を森で表現する
- 各位置 i は参照先 $R(i)$ を持ちうる
- 参照先がない位置 v は長さ 1 の項になる
- 一方で、v に参照先 w があると、w は v の親になる

文字列長さ n に対して、ASP 入力は n の文字

例

入力 T := abaaababa にとして、

```
#const
n=10.
t(1,97).
t(2,98).
t(3,97).
t(4,97).
t(5,97).
t(6,98).
t(7,97).
t(8,98).
t(9,97).
```

BMS の ASP 符号化

ASP コード:

```
{r(I,J);r(J,I)}1 :- t(I,C),t(J,C),I<J.
:- not { r(I,_) } 1, I=1..n.
p(I) :- 0 { r(I,_) } 0, I=1..n.
p(I) :- r(I,J), not r(I-1,J-1).
#edge (I,J): r(I,J).
#minimize { 1,I : p(I) }.
#show r/2. #show p/1.
```

BMS の ASP 符号化

ASP コード:

```
{r(I,J);r(J,I)}1 :- t(I,C),t(J,C),I<J.
:- not { r(I,_) } 1, I=1..n.
p(I) :- 0 { r(I,_) } 0, I=1..n.
p(I) :- r(I,J), not r(I-1,J-1).
#edge (I,J): r(I,J).
#minimize { 1,I : p(I) }.
#show r/2. #show p/1.
```

1. 写像 R を変数 r で表現する
2. $R(i) \leftarrow j$ または $R(j) \leftarrow i$ 可能性があるが、同時に二つはできない

$$\forall i,j \in [1..n], i < j \wedge T[i] = T[j] : r_{i,j} + r_{j,i} \leq 1 \quad \text{(BMS1)}$$

[Bannai+, 方程式 (9)]
$\{\mathcal{O}(n^2), \mathcal{O}(1)\}$

BMS の ASP 符号化

ASP コード:

```
{r(I,J);r(J,I)}1 :- t(I,C),t(J,C),I<J.
:- not { r(I,_) } 1, I=1..n.
p(I) :- 0 { r(I,_) } 0, I=1..n.
p(I) :- r(I,J), not r(I-1,J-1).
#edge (I,J): r(I,J).
#minimize { 1,I : p(I) }.
#show r/2. #show p/1.
```

2. $R(i)$ は一つ値以下を表現する

$$\forall i \in [1..n] : \sum_{j=1}^{n} r_{i,j} \leq 1 \qquad \text{(BMS2)}$$

[Bannai+, 方程式 (11)]

$\{\mathcal{O}(n), \mathcal{O}(1)\}$

BMS の ASP 符号化

ASP コード:

```
{r(I,J);r(J,I)}1 :- t(I,C),t(J,C),I<J.
:- not { r(I,_) } 1, I=1..n.
p(I) :- 0 { r(I,_) } 0, I=1..n.
p(I) :- r(I,J), not r(I-1,J-1).
#edge (I,J): r(I,J).
#minimize { 1,I : p(I) }.
#show r/2. #show p/1.
```

3. $T[i]$ は参照を持たない $\Rightarrow$ $T[i]$ は項の開始位置

$$\sum_{j=1}^{n} r_{i,j} = 0 \implies p_i \quad \text{(BMS3)}$$

[Bannai+, 方程式 (13)]
$\{\mathcal{O}(n), \mathcal{O}(1)\}$

BMS の ASP 符号化

ASP コード:

```
{r(I,J);r(J,I)}1 :- t(I,C),t(J,C),I<J.
:- not { r(I,_) } 1, I=1..n.
p(I) :- 0 { r(I,_) } 0, I=1..n.
p(I) :- r(I,J), not r(I-1,J-1).
#edge (I,J): r(I,J).
#minimize { 1,I : p(I) }.
#show r/2. #show p/1.
```

4. $T[i]$ の参照を左へ拡張できない $\Rightarrow$ $T[i]$ は項の開始位置

$$\forall i,j \in [1..n]:$$

$$r_{i,j} \land \neg r_{i-1,j-1} \implies p_i \quad \text{(BMS4)}$$

[Bannai+, 方程式 (15)]
$\{\mathcal{O}(n^2), \mathcal{O}(1)\}$

BMS の ASP 符号化

ASP コード:

```
{r(I,J);r(J,I)}1 :- t(I,C),t(J,C),I<J.
:- not { r(I,_) } 1, I=1..n.
p(I) :- 0 { r(I,_) } 0, I=1..n.
p(I) :- r(I,J), not r(I-1,J-1).
#edge (I,J): r(I,J).
#minimize { 1,I : p(I) }.
#show r/2. #show p/1.
```

5. ASP の魔法： r は閉路を持たないかどうか確認

方針

Bannai+'22: 閉路の確認を推移閉包で行う：

$$\forall i \in [1..n] : r_{i,j} \Rightarrow c_{i,j}$$

$$\{\mathcal{O}(n^2), \mathcal{O}(1)\}$$

$$\forall i, j, k \in [1..n] : c_{i,j} \wedge r_{j,k} \Rightarrow c_{i,k}$$

$$\{\mathcal{O}(n^3), \mathcal{O}(1)\}$$

$$\forall i, j \in [1..n] : c_{i,j} \Rightarrow \neg c_{j,i}$$

$$\{\mathcal{O}(n^2), \mathcal{O}(1)\}$$

BMS の ASP 符号化

ASP コード:

```
{r(I,J);r(J,I)}1 :- t(I,C),t(J,C),I<J.
:- not { r(I,_) } 1, I=1..n.
p(I) :- 0 { r(I,_) } 0, I=1..n.
p(I) :- r(I,J), not r(I-1,J-1).
#edge (I,J): r(I,J).
#minimize { 1,I : p(I) }.
#show r/2. #show p/1.
```

6. 項の開始位置を最小化 ⇔ 項の個数を最小化

$$\sum_{i=1}^{n} p_i \text{ を最小化} \qquad \text{(MINP)}$$

BMS の ASP 符号化

ASP コード:

```
{r(I,J);r(J,I)}1 :- t(I,C),t(J,C),I<J.
:- not { r(I,_) } 1, I=1..n.
p(I) :- 0 { r(I,_) } 0, I=1..n.
p(I) :- r(I,J), not r(I-1,J-1).
#edge (I,J): r(I,J).
#minimize { 1,I : p(I) }.
#show r/2. #show p/1.
```

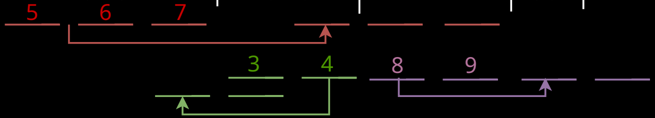
7. p_i と r_i の値を出力

例

clingo の出力:

```
p(1)
p(4)
p(6)
p(8)
p(9)
r(1,5)
r(2,6)
r(3,7)
r(4,3)
r(5,4)
r(6,8)
r(7,9)
```

	1	2	3		4	5		6	7		8		9
$T =$	a	b	a	|	a	a	|	b	a	|	b	|	a
		(5, 7)			(3, 4)						(b)		(a)



ファイル		計算時間 (秒)	
名前	n	SAT	ASP
BIB	128	0.28	0.02
BIB	256	44.13	0.49
BOOK1	128	0.22	0.02
BOOK1	256	N/A	0.13
BOOK2	128	0.55	0.02
BOOK2	256	N/A	0.14
NEWS	128	0.15	0.02
NEWS	256	13.61	0.27
OBJ2	128	3.64	0.04
OBJ2	256	N/A	19.89
PAPER1	128	0.56	0.09
PAPER1	256	25.18	0.12
PAPER2	128	0.26	0.27
PAPER2	256	29.78	27.29
PAPER4	128	0.19	0.04
PAPER4	256	31.75	0.46
PROGC	128	0.44	0.02
PROGC	256	N/A	0.21
PROGL	128	18.23	0.15
PROGL	256	N/A	39.84
PROGP	128	0.24	0.02
PROGP	256	N/A	1.10
TRANS	128	0.45	0.03
TRANS	256	N/A	2.65

実験とまとめ

- SAT は pysat の RC2 を利用する
- 計算機はノートパソコン (Intel i5-1135G7)

なぜ ASP のほうが速い？

- #edge は ASP の定理証明器で高速に計算できる
- 余分の節を省いた

ファイル		変数の個数	
名前	n	SAT	ASP
BIB	128	27 001	926
BIB	256	204 806	3546
BOOK1	128	21 743	790
BOOK1	256	N/A	3500
BOOK2	128	28 822	1004
BOOK2	256	N/A	3608
NEWS	128	13 923	842
NEWS	256	101 441	3114
OBJ2	128	235 960	3808
OBJ2	256	N/A	18 982
PAPER1	128	29 137	1168
PAPER1	256	124 272	3220
PAPER2	128	25 046	1040
PAPER2	256	134 968	3290
PAPER4	128	15 696	836
PAPER4	256	131 283	3216
PROGC	128	37 116	1110
PROGC	256	N/A	4014
PROGL	128	1 730 977	17 596
PROGL	256	N/A	42 112
PROGP	128	19 834	1000
PROGP	256	N/A	4154
TRANS	128	26 657	918
TRANS	256	N/A	3652

実験とまとめ

- SAT は pysat の RC2 を利用する
- 計算機はノートパソコン (Intel i5-1135G7)

なぜ ASP のほうが速い？

- #edge は ASP の定理証明器で高速に計算できる
- 余分の節を省いた

ファイル		節の個数	
名前	n	SAT	ASP
BIB	128	83 132	267
BIB	256	670 873	633
BOOK1	128	65 853	237
BOOK1	256	N/A	636
BOOK2	128	89 344	270
BOOK2	256	N/A	654
NEWS	128	41 689	279
NEWS	256	327 701	618
OBJ2	128	794 509	240
OBJ2	256	N/A	639
PAPER1	128	91 183	237
PAPER1	256	401 521	642
PAPER2	128	77 254	234
PAPER2	256	436 216	633
PAPER4	128	47 242	237
PAPER4	256	425 296	639
PROGC	128	116 631	291
PROGC	256	N/A	633
PROGL	128	5 982 279	315
PROGL	256	N/A	717
PROGP	128	60 868	294
PROGP	256	N/A	714
TRANS	128	81 589	255
TRANS	256	N/A	636

実験とまとめ

- SAT は pysat の RC2 を利用する
- 計算機はノートパソコン (Intel i5-1135G7)

なぜ ASP のほうが速い？

- #edge は ASP の定理証明器で高速に計算できる
- 余分の節を省いた

ご清聴ありがとうございました

ASP について

解集合プログラミング

SAT 技術の広がり

問題を SAT に変換し SAT ソルバーを用いて解く手法が様々な分野で成功し，SAT 技術が大きな広がりを見せている

- 有界モデル検査 Biere'09
- Intel core I7 プロセッサ設計 Kaivola+'09
- Windows 7 デバイス・ドライバー検証 De Moura+'10 (SMT ソルバー Z3)
- ソフトウェア要素の依存性解析
- Eclipse Le Berre and Rapicault'09
- Fedora Linux (および RedHat, CentOS) の dnf
- プランニング (SATPLAN, Blackbox) Kautz+'92
- ショップ・スケジューリング Crawford+'94 田村+'09
- 解集合プログラミング (clingo) Gebser+'12
- 制約充足問題 (Sugar) 田村+'09

解集合プログラミング (ASP)

ASP は比較的新しい宣言的プログラミングパラダイムの一つである

- ASP 言語は一階論理に基づく知識表現言語の一種
- ASP システムは安定モデル意味論 Gelfond,Lifschitz'88 に基づく解集合を計算するシステム
- ASP の起源
- 演繹データベース
- 論理プログラミング (否定付き)
- 知識表現 & 非単調推論
- 制約充足 (特に，SAT)

ASP の応用

近年，SAT 技術を応用した高速な ASP システムが開発され，人工知能分野への実用的応用が急速に拡大している

- ASP システム: clingo, WASP Web, DLV, etc.
- 応用
  - ロボット工学，システム生物学，モデル検査，
  - マルチエージェント，チーム編成，テストケース生成，
  - プランニング，スケジューリング, etc.
- 国際会議: IJCAI, AAAI, ICLP, KR, LPNMR, etc.