HOKKAIDO UNIVERSITY

Title	VF符号に対する圧縮パターン照合の性能について
Author(s)	吉田, 諭史; 喜田, 拓也
Description	ERATO湊離散構造処理系プロジェクト : 2010年度初冬のワークショップ（ERATO合宿）. 2010年11月29日（月）～12月1日（水）. 札幌北広島クラッセホテル.
Relation	2010年度科学技術振興機構ERATO湊離散構造処理系プロジェクト講究録. p.401-403.
Issue Date	2011-06
Doc URL	https://hdl.handle.net/2115/48354
Type	conference presentation
File Information	04.06_kida.pdf

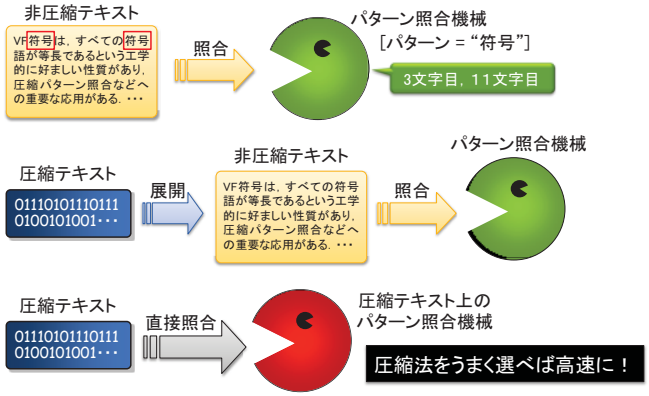
Hokkaido University Collection of Scholarly and Academic Papers : HUSCAP

北海道大学 Hokkaido University

VF符号に対する圧縮パターン照合の性能について

北海道大学 大学院情報科学研究科
吉田諭史、喜田拓也

研究の背景:圧縮パターン照合問題(CPM問題)



研究の背景:CPM問題に適したデータ圧縮

- 圧縮パターン照合に適した圧縮法の特徴
  - 符号語の区切りが明確である
    - 特にバイト単位や固定長の符号語であることが望ましい
  - 静的でコンパクトな辞書を用いる
    - 動的に辞書が変化すると，テキストスキャン時に余計な時間がかかる
  - (その上でなるべく)圧縮率が良いこと
    - 処理すべきデータ量が削減される

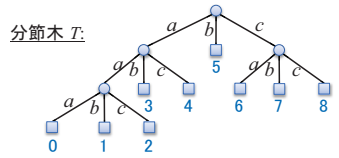
すべての符号語が一定の長さ(固定長)

VF符号
(Variable-length-to-Fixed-length Code)

- Tunstall符号(Tunstall 1967)
- AIVF符号(Yamamoto&Yokoo 2001)
- STVF符号(Klien&Shapira2008, Kida2009)

(分節木を用いた)VF符号

- 入力テキストを，分節木と呼ばれる木構造を用いて可変長のブロックに分割し，それぞれに固定長の符号化を割り当てることで符号化を行う
  - 分節木の各ノードは文字列に対応している
  - 文字列が辞書(分節木)に登録されており，それぞれに固定長の符号語が割り当てられていると考えて良い



入力テキスト:	a	b	b	a	a	b	b	a	a	a	c	a	c	c
符号化テキスト:	3		5		1		5		0		6		8	

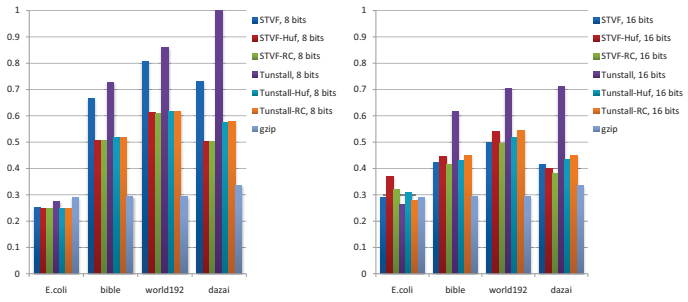
今回の話

- 実際のVF符号上での圧縮パターン照合の性能は？
  - gzipファイルに対してzgrepするのと比べてどうなの？
  - 非圧縮テキストに対するfgrepと比べてどうなの？
- VF符号語列をさらにエントロピー符号化すれば，圧縮率は向上するか？
  - (前に話した)分節木の繰り返し学習するよりも，手っ取り早く圧縮性能を上げられるのでは？(gzip並みにできる？)
- 符号長が短いとき(8bit)と長いとき(16bit)で性能は変化するか？
  - 分節木が小さい → 圧縮率低い，照合・展開処理は楽
  - 分節木が大きい → 圧縮率高い，照合・展開処理は大変？

実験

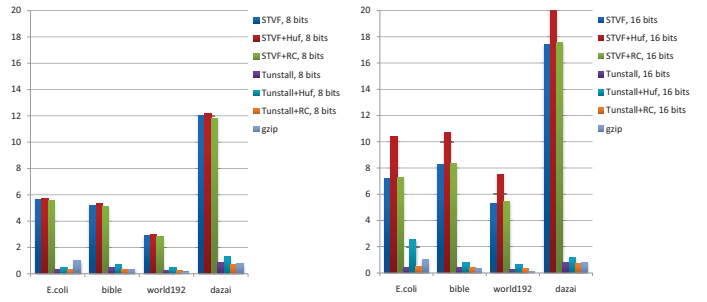
- Tunstall符号とSTVF符号[KidaDCC2009]の各種性能(圧縮率，圧縮速度，展開速度，パターン照合速度)を比較した
  - 単独で用いる(符号語長 8bit，16bit)
  - エントロピー符号化(Huffman符号，RangeCoder)と組み合わせる(符号語長 8bit，16bit)
  - 比較手法:
    - zgrep : gzipされたテキストを一度展開してからgrepする
    - fgrep : 非圧縮テキストを高速に(BM型アルゴリズムで)grepする
- VF符号上でのパターン照合アルゴリズムは，[Kida2003]のcollage systemに基づくprefix型アルゴリズムを採用($O(n+m^2)$時間，$O(D+m^2)$領域)
  - ただし，実装の簡便さから前処理$O(m^3)$のアルゴリズムを採用

圧縮率（圧縮後ファイルサイズ/元ファイルサイズ）



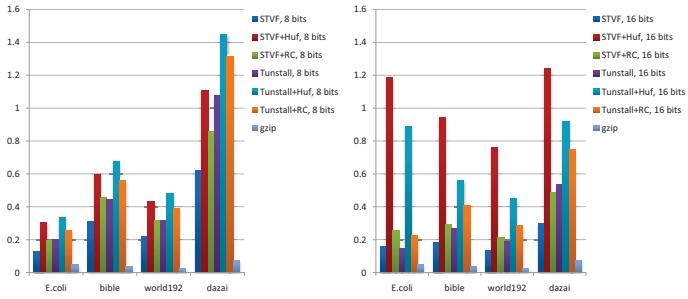
E.coli: 4,638,690 byte, Σ = 4, E.coliバクテリアのゲノムデータ
bible: 4,047,392 byte, Σ = 63 , King James版聖書
world192: 2,473,400 byte, Σ = 94 , The CIA world fact book
dazai: 7,268,943 byte, Σ = 141 ,太宰治全集(UTF-8)

圧縮速度（CPU time by time command（秒））



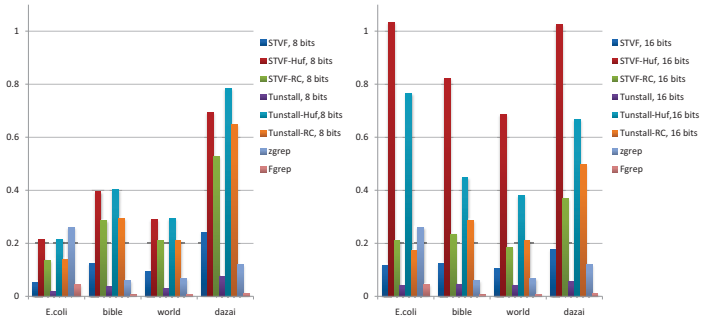
C++ implementation, compiled by g++ GNU version 3.4
Intel Xeon processor 3.00GHz dual core hyper threading with 12GB memory
Red Hat Enterprise Linux ES release 4.

展開速度（CPU time by time command（秒））



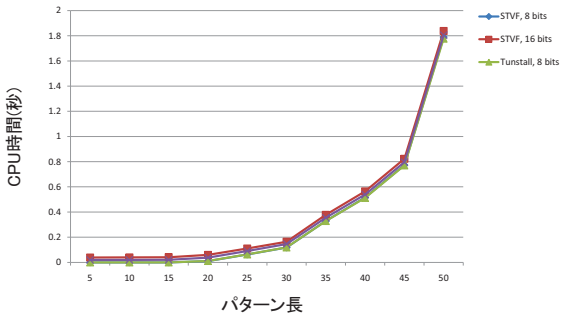
C++ implementation, compiled by g++ GNU version 3.4
Intel Xeon processor 3.00GHz dual core hyper threading with 12GB memory
Red Hat Enterprise Linux ES release 4.

パターン照合（全体:CPU time by time command（秒））



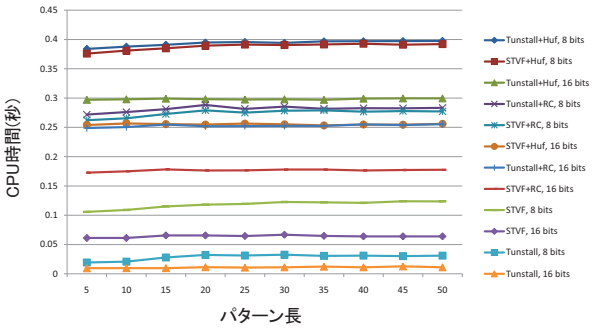
C++ implementation, compiled by g++ GNU version 3.4
Intel Xeon processor 3.00GHz dual core hyper threading with 12GB memory
Red Hat Enterprise Linux ES release 4.
パターン長は15

パターン照合（前処理時間; 使用テキストはbible）



C++ implementation, compiled by g++ GNU version 3.4
Intel Xeon processor 3.00GHz dual core hyper threading with 12GB memory
Red Hat Enterprise Linux ES release 4.
bible: 4,047,392 byte, Σ = 63 , King James版聖書

パターン照合（テキスト走査; 使用テキストはbible）



C++ implementation, compiled by g++ GNU version 3.4
Intel Xeon processor 3.00GHz dual core hyper threading with 12GB memory
Red Hat Enterprise Linux ES release 4.
bible: 4,047,392 byte, Σ = 63 , King James版聖書

まとめ

- VF符号上での圧縮パターン照合は，
  - 現状zgrepするよりは高速にできるが，fgrepほどではない
    - Prefix型アルゴリズムであることが一因（fgrepはSuffix型アルゴリズム）
    - 前処理に時間がかかる → パターン長に対するスケーラビリティが低い
- VF符号語列をさらにエントロピー符号化すると，
  - 圧縮率は向上するが，gzip並みとはいかなかった
  - エントロピー符号を展開する分，時間が余計にかかる
  - Huffman符号は，VF符号の符号長が長い場合には不向き
- 符号長が短いとき（8bit）と長いとき（16bit）とでは，
  - 短い(8bit)場合のほうが圧縮率は悪いものの，圧縮速度・パターン照合速度は速い（特にTunstall(8)が良かった）
  - ただし，展開速度は16bitのほうが速い．
    また，テキスト走査だけをみれば，16bitのほうが処理が速い．