Data Compression using Variable-to-Fixed Length Codes

Hokkaido University
Graduate school of Information Science and Technology, Division of Computer Science

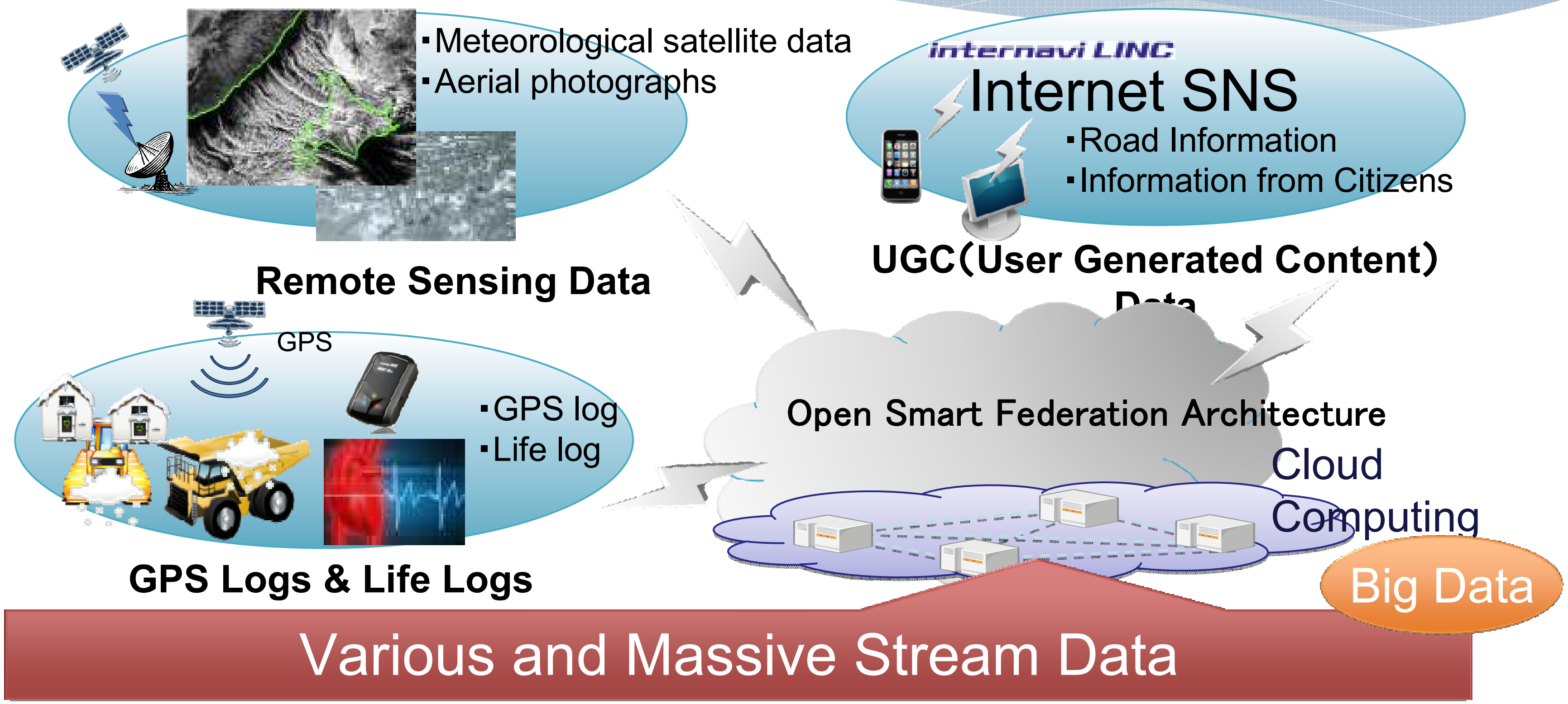
Takuya Kida
kida@ist.hokudai.ac.jp

with Satoshi Yoshida, Hirohito Sasakawa, and Kei Sekine

Contents

* Background and aim
  * Related-works
* Variable-to-Fixed length codes (VF codes)
* Grammar-based compression
  * Re-Pair algorithm
* Proposed method (Re-Pair VF codes)
* Experimental results
* Summary
* Biography

Background and aim

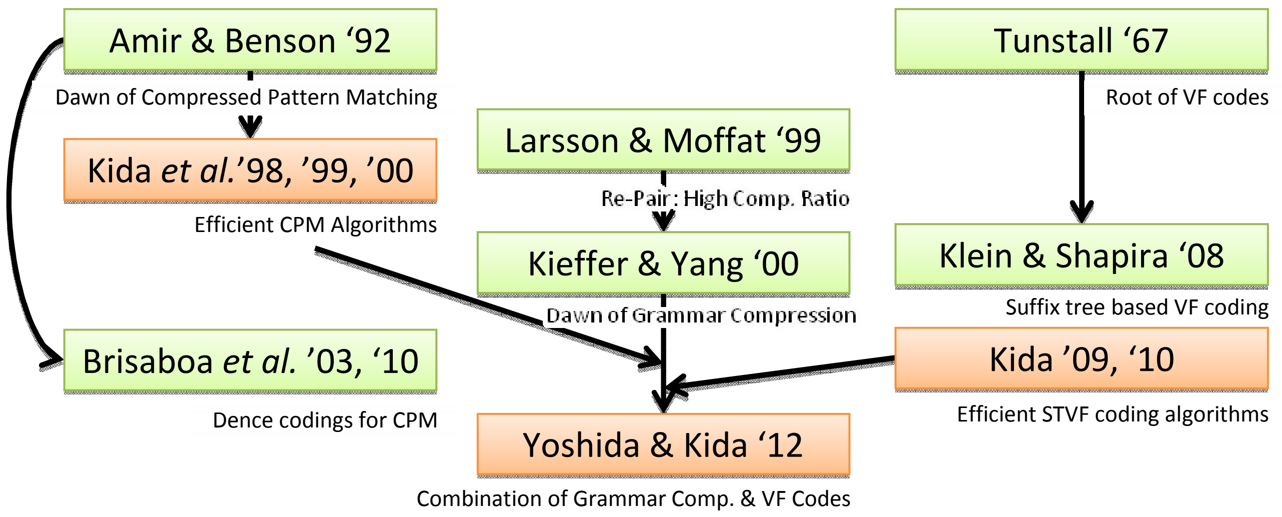


Each data entry isn't significant. The data are usually too redundant to store into an

Develop an accessible data compression whose compressed data are reusable for data searching and

Related works

* Problem: Existing methods are inconvenient for reusing the compressed data, because they must follow the decompression process.



* Originality: We have data compression techniques that can search keywords on compressed data directly without decompressing and reach a relatively good compression performance.

Variable-to-Fixed Length Codes

- Compression method that splits the input text into variable length substring and then converts them into fixed length codewords.
  - Strong point： Easy to handle the compressed data
    => Enables fast information retrieval or data mining
  - Weak point： Hard to get a good compression ratio

Grammar Compressions

model data very well

+

VF codes

provide high accessibility

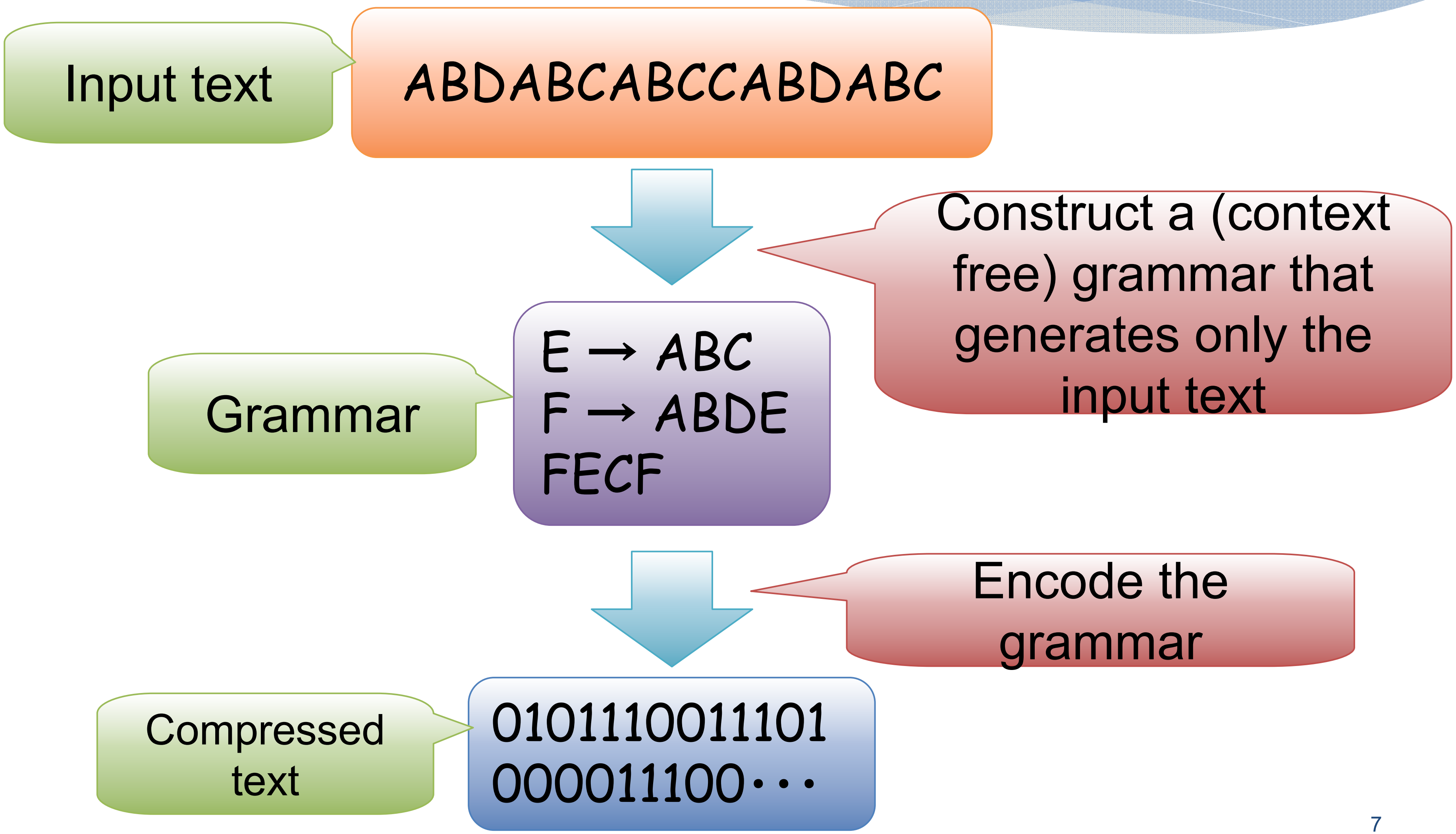
Satisfy both of high comp. ratio and accessibility!

Variations of codes

		Input Text	
		Fixed Length	Variable Length
Compressed Text	Fixed Length	**FF Code** (Fixed length to Fixed length code)	**VF Code** (Variable length to Fixed length code)
	Variable Length	**FV Code** (Fixed length to Variable length code)	**VV Code** (Variable length to Variable length code)

Tunstall Code

Grammar-based compression



Re-pair algorithm

Replace most frequent bigram into a new symbol

AAABACAAABCCAAAB

↓

DABACDABCCDAB

↓

EBACEBCCEB

↓

FACFCCF

↓

FAGCG

dictionary

D → AA

E → DA

F → EB

G → CF

Encode with
variable length code
(a variation of Huffman coding)

Proposed Method (Re-Pair VF codes [Kida&Yoshida DCC2013])

The basic idea

- In the Re-Pair algorithm, symbol replacement does not always imply improvement of compression ratio.
  ⇨We want to encode with the best intermediate dictionary, which gives highest compression ratio.
- We find the best dictionary and then expand the rules that do not included in it when translating to a grammar.
- Finally, we encode the obtained grammar by Fixed length codes.

How to find the best?

Dict.

s :
- D → AA
- E → DA
- F → EB
- G → CF
- H → GE
- I → FF
- J → IG
- K → HI

AADAEBCFGEFFIGHI EIKJBCEFICEK ($2s+n$)

The number of sort of symbols: $s+|\Sigma|$

We need $\lceil \log(s+|\Sigma|) \rceil$ bits for each symbol.

sum: $(2s+n)\lceil \log(s+|\Sigma|) \rceil$ bits

Calculate this value for each substitution to store the size of dictionary that gives the highest compression ratio.

Compressed sequence:

EIKJBCEFICEK (n)

Rewind the dictionary

Compressed sequence:
EIKJBCEFICEK

Dictionary
- D → AA
- E → DA
- F → EB
- G → CF
- H → GE
- I → FF
- J → IG
- K → HI

expand this part

Compressed sequence by the best dictionary:
EFFGEFFFFFFGBCEFFFCEGEFF

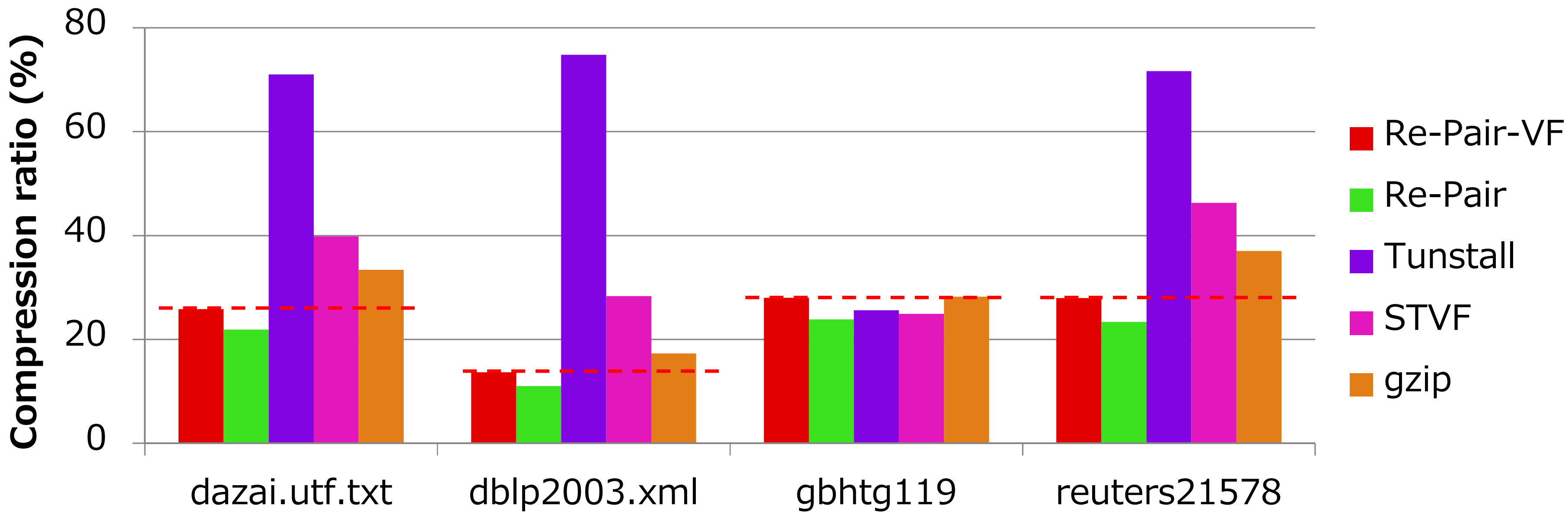
AADAEBCF EFFGEFFFFFFGBCEFFFCEGEFF

Experiments

We compared compression ratios, compression times, decompression times, and pattern matching speeds among these methods.

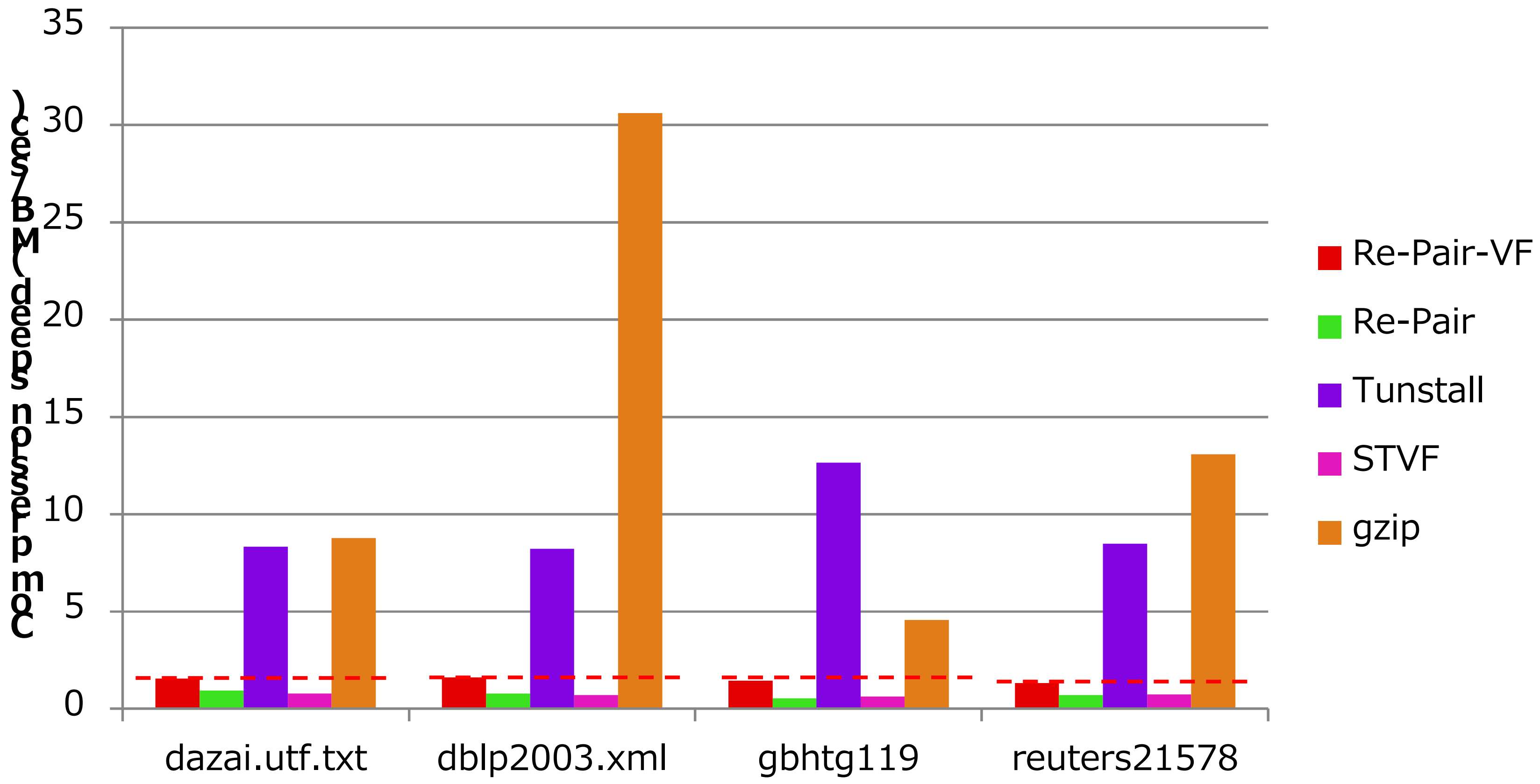
- Method
  - VF code via Re-pair (Re-Pair-VF)
  - Original Re-pair (Re-Pair)
  - Tunstall code
  - STVF code (proposed by Kida)
  - gzip (zgrep which decompress the data then search keywords)
- Data
  - dazai.utf.txt (Japanese text (UTF-8 encoded), 7MB)
  - dblp2003.xml (XML data, 90MB)
  - gbhtg119 (DNA data, 87MB)
  - reuters21578 (English text, 19MB)

Compression ratio
[Yoshida&Kida2013]

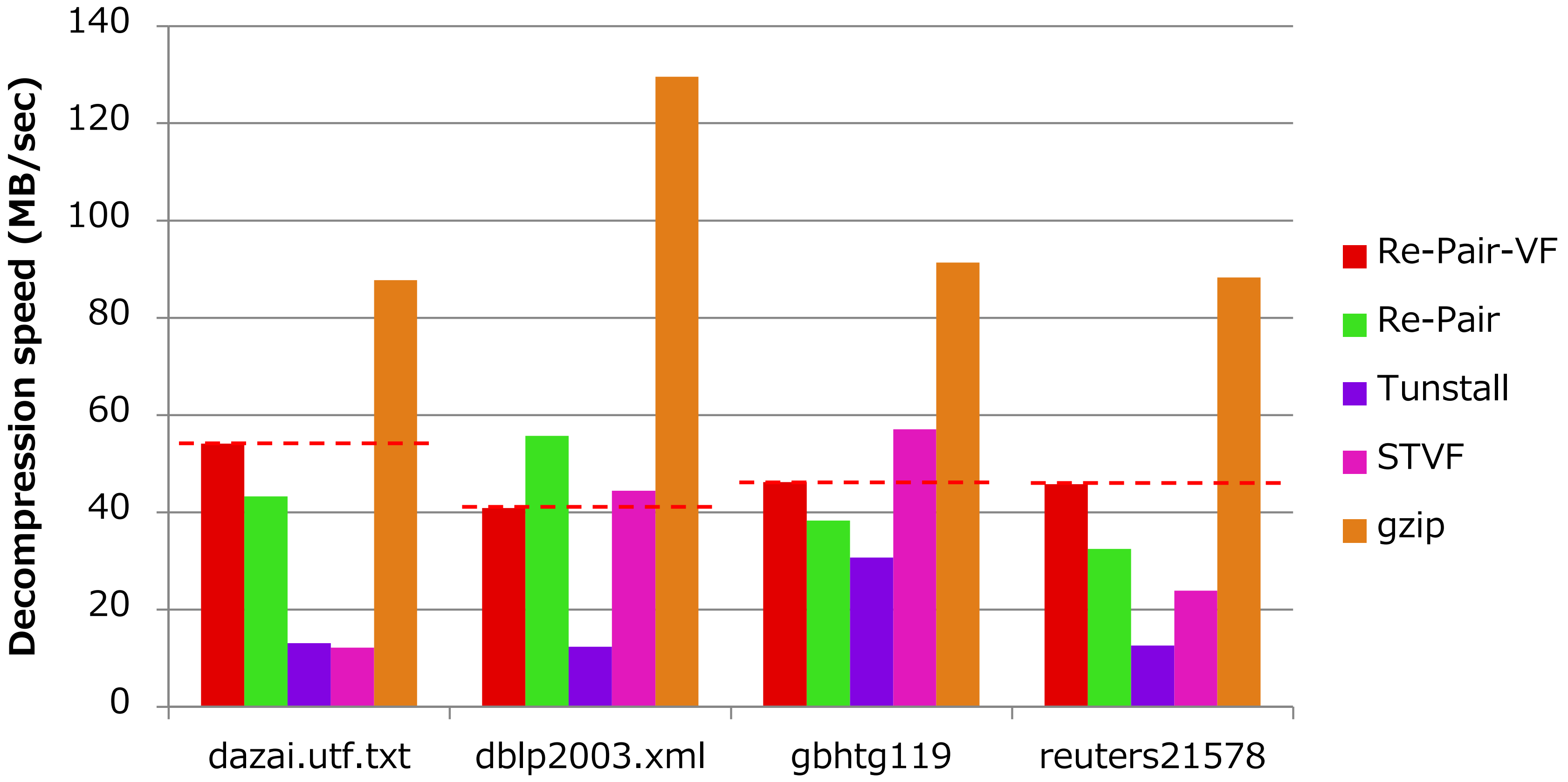


Re-Pair-VF overcomes gzip!

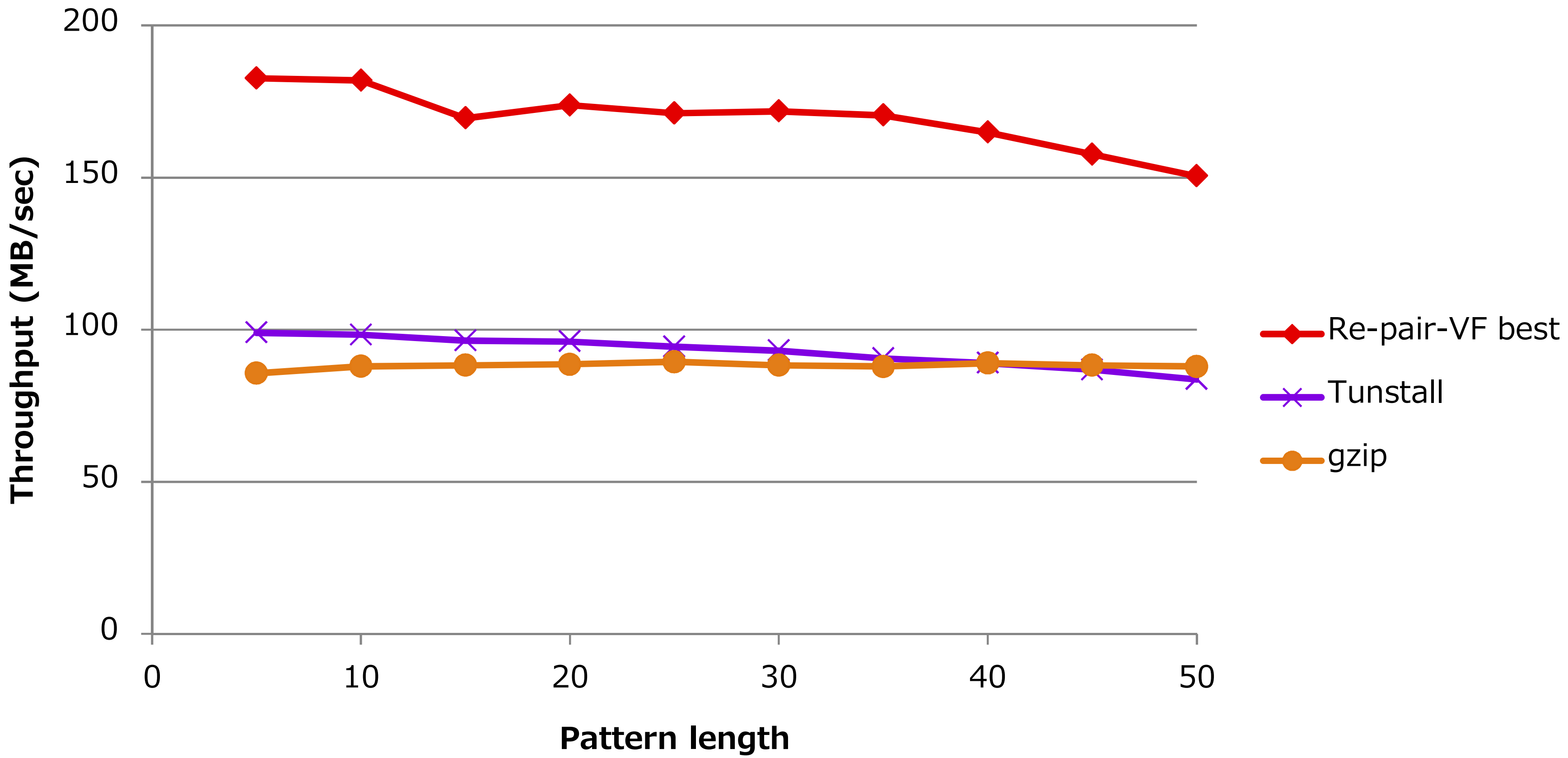
Compression speed
[Yoshida&Kida2013]



Decompression speed
[Yoshida&Kida2013]

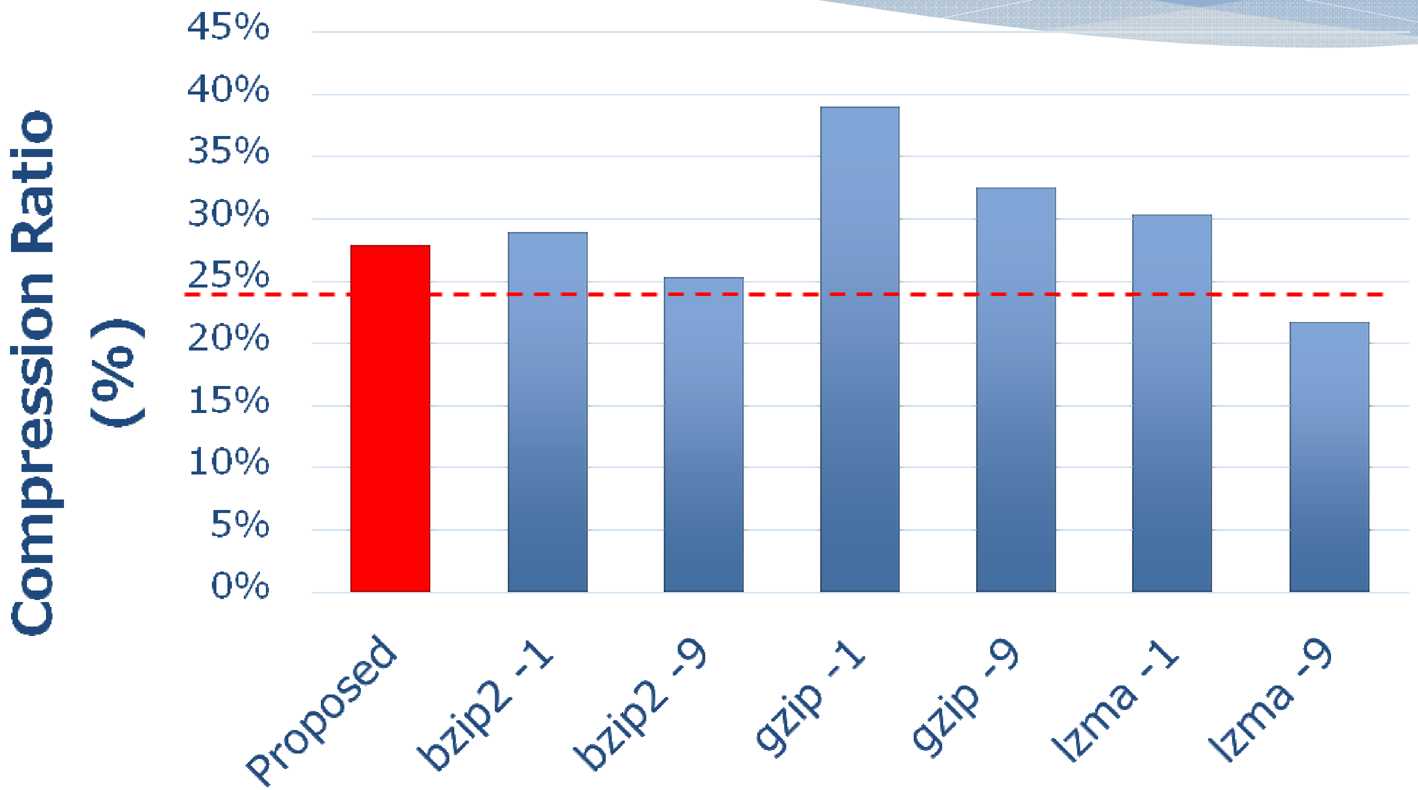


Search speed [Yoshida&Kida2013]
(for reuters21578[English news paper])



$$\text{Throughput} = \frac{\text{Original text length}}{\text{Pattern matching time}}$$

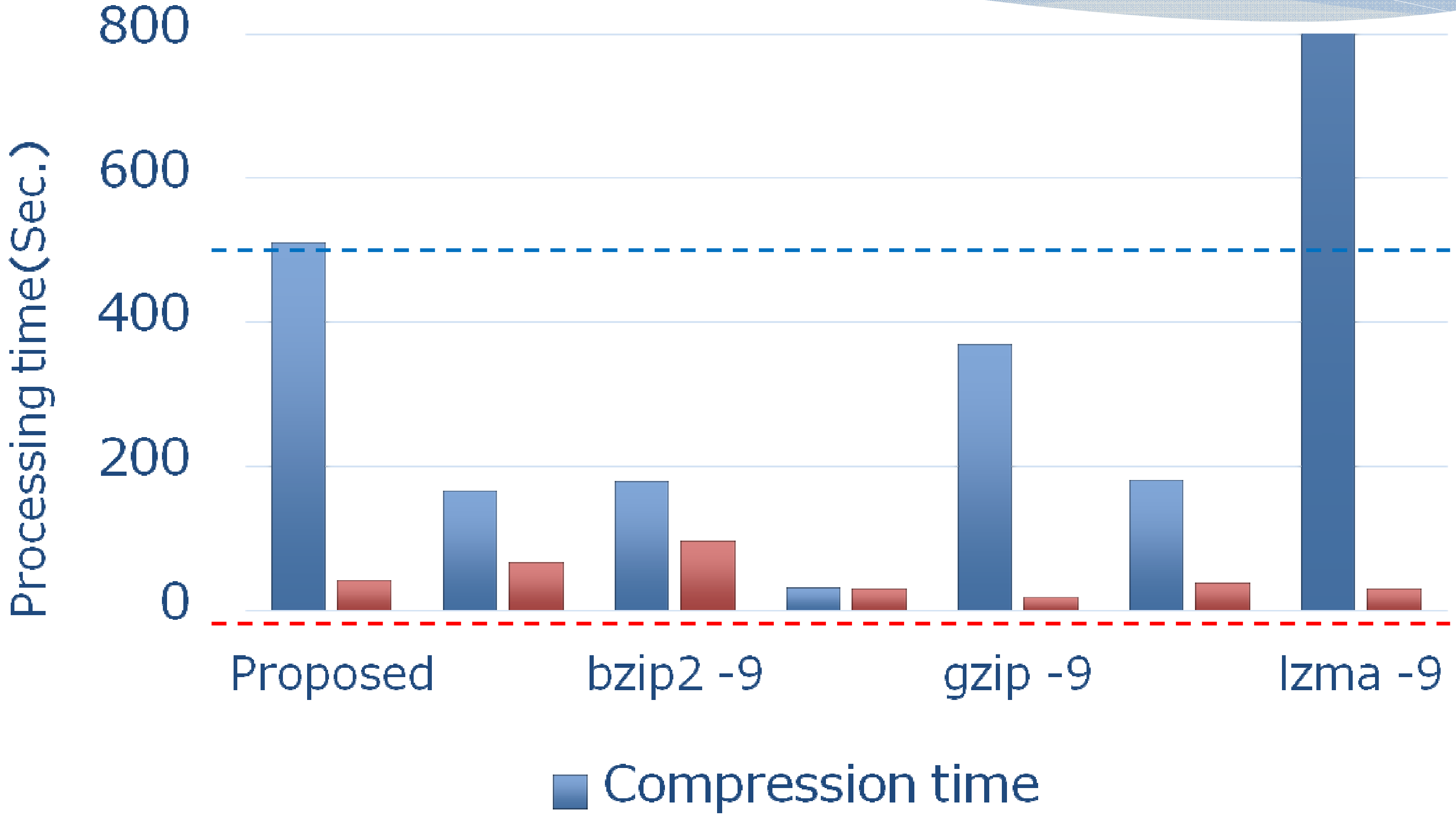
Application to large data [Sekine *etal*.2013] (English 2.2GB text from Pizza&Chili corpus)



Our proposed method achieved a good compression ratio comparable to bzip2 (about 27%) for large texts

(codeword length 19bit, block size 128MB, ratio of shared dictionary 3/8, sampling text size 128MB)

Application to large data [Sekine etal.2013]
(English 2.2GB text from Pizza&Chili corpus)



The proposed method takes much time for compressing,
but it can decompress at high speed!

Summary

* We focused attention to VF codes, which are useful for reusing compressed data, and developed an efficient VF coding based on Re-pair algorithm.
* The coding enables us to do fast keyword search, in addition to make partial decompression and re-compression easy.

Future works:

* Improvement of compression speed
* Realizing pinpoint access to compressed data with an original text position.
* Development an online coding algorithm

Thank you for your kind attention!

Biography

Takuya Kida received the B.S. degree in Physics, the M.S. and Dr. (Information Science) degree all from Kyushu University in 1997, 1999, and 2001, respectively. He was a Full-time Lecturer of Kyushu University Library from October 2001 to March 2004. He is currently an Associate Professor of Division of Computer Science, Graduate School of Information Science and Technology, Hokkaido University, since April 2004. His research interests include Text Algorithms and Data Structures, Information Retrieval, and Data Compression. He is a member of IEICE, IPSJ, and DBSJ.