Re-Pair Re-Visited

Francisco Claude, Rodrigo González, Gonzalo Navarro, Luís Russo

U. Chile, INESC Lisboa

Nov 1, 2007

Outline

1. Introduction
2. Compressing with Re-Pair
   - Exact Version
   - Approximate Version
3. Dictionary Compression
4. Local Decompression
5. Theoretical Analysis
6. Experimental Results
7. Conclusions and Future Work

Outline

1. Introduction
2. Compressing with Re-Pair
   - Exact Version
   - Approximate Version
3. Dictionary Compression
4. Local Decompression
5. Theoretical Analysis
6. Experimental Results
7. Conclusions and Future Work

Outline

1. Introduction
2. Compressing with Re-Pair
   - Exact Version
   - Approximate Version
3. Dictionary Compression
4. Local Decompression
5. Theoretical Analysis
6. Experimental Results
7. Conclusions and Future Work

Outline

1. Introduction
2. Compressing with Re-Pair
   - Exact Version
   - Approximate Version
3. Dictionary Compression
4. Local Decompression
5. Theoretical Analysis
6. Experimental Results
7. Conclusions and Future Work

Outline

1. Introduction
2. Compressing with Re-Pair
   - Exact Version
   - Approximate Version
3. Dictionary Compression
4. Local Decompression
5. Theoretical Analysis
6. Experimental Results
7. Conclusions and Future Work

Outline

1. Introduction
2. Compressing with Re-Pair
   - Exact Version
   - Approximate Version
3. Dictionary Compression
4. Local Decompression
5. Theoretical Analysis
6. Experimental Results
7. Conclusions and Future Work

Outline

1. Introduction
2. Compressing with Re-Pair
   - Exact Version
   - Approximate Version
3. Dictionary Compression
4. Local Decompression
5. Theoretical Analysis
6. Experimental Results
7. Conclusions and Future Work

Outline

1. Introduction
2. Compressing with Re-Pair
   - Exact Version
   - Approximate Version
3. Dictionary Compression
4. Local Decompression
5. Theoretical Analysis
6. Experimental Results
7. Conclusions and Future Work

Introduction

Methods

- Statistical
- Dictionary based

Differences

- Statistical methods usually achieve better compression ratio
- Dictionary based allow faster decompression

Statistical

- Huffman
- PPM

Dictionary based

- LZ77/78
- Re-Pair (Larsson & Moffat 1999)

Applications

Successful applications of Re-Pair

- Phrase-based compression and browsing (Moffat & Wan 2003)
- Compressing suffixs arrays (González & N. 2006)
- Web Graph compression (Claude & N. 2007)

Other applications?

- We study the application of Re-Pair to normal text files
- We want to exploit its efficient local decompression
- Is there any guarantee on the compression ratio achieved?

Extending Re-Pair

Re-Pair

- New algorithms for compressing with Re-Pair
- New dictionary compression
- Experimental study of Re-Pair
- Theoretical analysis

Results

- New exact and approximate compression methods
- The dictionary is reduced to 50% of its size
- The dictionary is a compact data structure
- Re-Pair achieves good compression ratios
- It is very fast for decompressing
- Re-Pair is able to extract snippets from disk very efficiently
- It achieves $2nH_k + o(n \log \sigma)$ bits of space

Re-Pair

Re-Pair

- Find the most frequent pair ab in T
- Create the rule $s \rightarrow ab$
- Replace occurrences of ab by s
- Iterate until every pair appears only once

Original Re-Pair algorithm

- Very fast, $O(n)$
- Consumes much memory ($20n$ bytes)

Outline

1. Introduction
2. Compressing with Re-Pair
   - Exact Version
   - Approximate Version
3. Dictionary Compression
4. Local Decompression
5. Theoretical Analysis
6. Experimental Results
7. Conclusions and Future Work

Re-Pair

$a\ a\ a\ b\ c\ a\ a\ b\ a\ a\ a\ b\ c\ a\ b\ d\ a\ b\ d$

a	a	4
a	b	5
b	c	2
c	a	2
b	d	2
d	a	1

$A \longrightarrow a\ b$

$a\ a\ A\ c\ a\ A\ a\ a\ A\ c\ A\ d\ A\ d$

Re-Pair

$a\ a\ A\ c\ a\ A\ a\ a\ A\ c\ A\ d\ A\ d$

$a\ a$	2	$A \longrightarrow$	$a\ b$
$a\ A$	3	$B \longrightarrow$	$a\ A$
$A\ c$	2		
$c\ a$	1		
$A\ d$	2		
$c\ A$	1		
$d\ A$	1		

$a\ B\ c\ B\ a\ B\ c\ A\ d\ A\ d$

Re-Pair

$a\ B\ c\ B\ a\ B\ c\ A\ d\ A\ d$

$a\ B$	2
$B\ c$	2
$c\ B$	1
$B\ a$	1
$c\ A$	1
$A\ d$	2
$d\ A$	1

$A \longrightarrow a\ b$

$B \longrightarrow a\ A$

$C \longrightarrow A\ d$

$a\ B\ c\ B\ a\ B\ c\ C\ C$

Re-Pair

$a\ B\ c\ B\ a\ B\ c\ C\ C$

$a\ B$	2
$B\ c$	2
$c\ B$	1
$B\ a$	1
$c\ C$	1
$C\ C$	1

$A \longrightarrow a\ b$

$B \longrightarrow a\ A$

$C \longrightarrow A\ d$

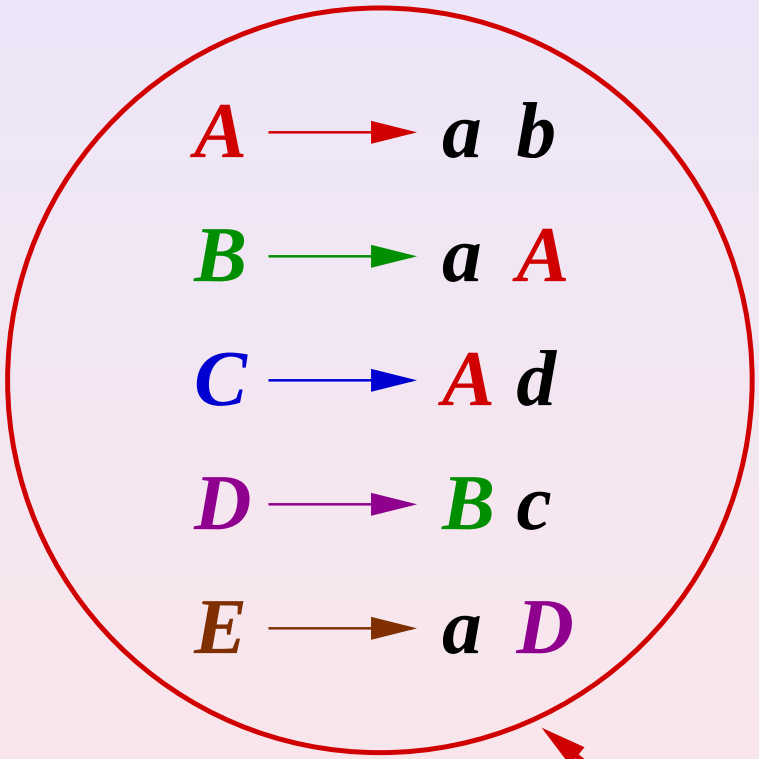
$D \longrightarrow B\ c$

$a\ D\ B\ a\ D\ C\ C$

Re-Pair

$a\ D\ B\ a\ D\ C\ C$

$a\ D$	2
$D\ B$	1
$B\ a$	1
$D\ C$	1
$C\ C$	1



dictionary

$E\ B\ E\ C\ C$ ← *compressed sequence*

Outline

1. Introduction
2. Compressing with Re-Pair
   - Exact Version
   - Approximate Version
3. Dictionary Compression
4. Local Decompression
5. Theoretical Analysis
6. Experimental Results
7. Conclusions and Future Work

General Idea - Exact Version

- The frequency of pairs is tracked with a Treap
- Every replacement modifies the Treap
- Every pair points to its next occurrence
- It achieves $O(n \log n)$ time

Advantages/Disadvantages

- Less memory usage
- Slower compression
- It is an alternative to the exact method

Outline

1. Introduction
2. Compressing with Re-Pair
   - Exact Version
   - Approximate Version
3. Dictionary Compression
4. Local Decompression
5. Theoretical Analysis
6. Experimental Results
7. Conclusions and Future Work

General Idea - Approximate Version

- We place the text in a larger array
- The space left is used to count with a hash table
- We select k pairs for replacing
- We replace them in one sweep over the text
- The space left in the array is used to track pairs

Advantages/Disadvantages

- Faster compression
- Tunable usage of memory
- Loses some compression

Outline

1. Introduction
2. Compressing with Re-Pair
   - Exact Version
   - Approximate Version
3. Dictionary Compression
4. Local Decompression
5. Theoretical Analysis
6. Experimental Results
7. Conclusions and Future Work

Improving Dictionary Representation

- Represent the dictionary using compact data structures
- We reduce up to 50% the space for the dictionary
- Especially important for secondary memory

Improving Dictionary Representation

Rules

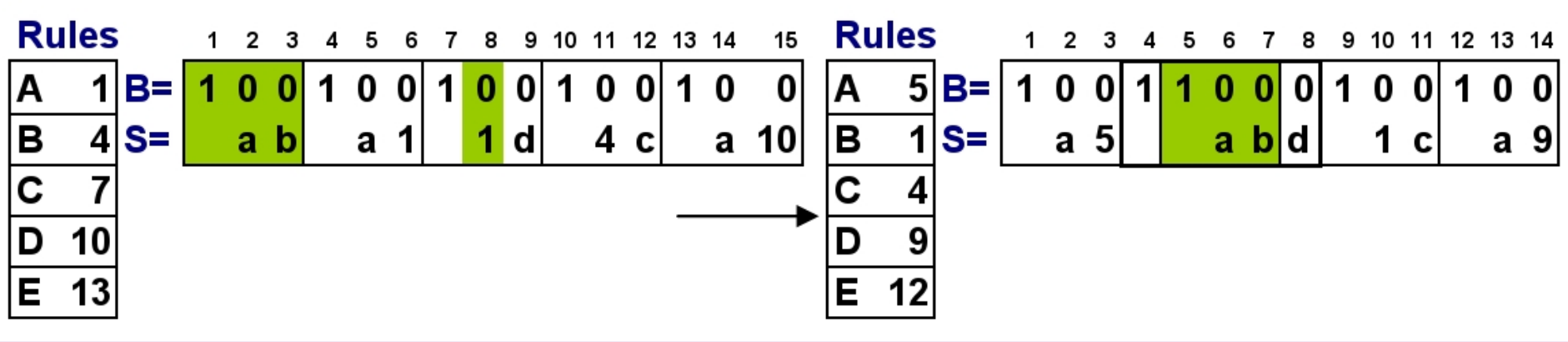
A	→	a	b
B	→	a	A
C	→	A	d
D	→	B	c
E	→	a	D

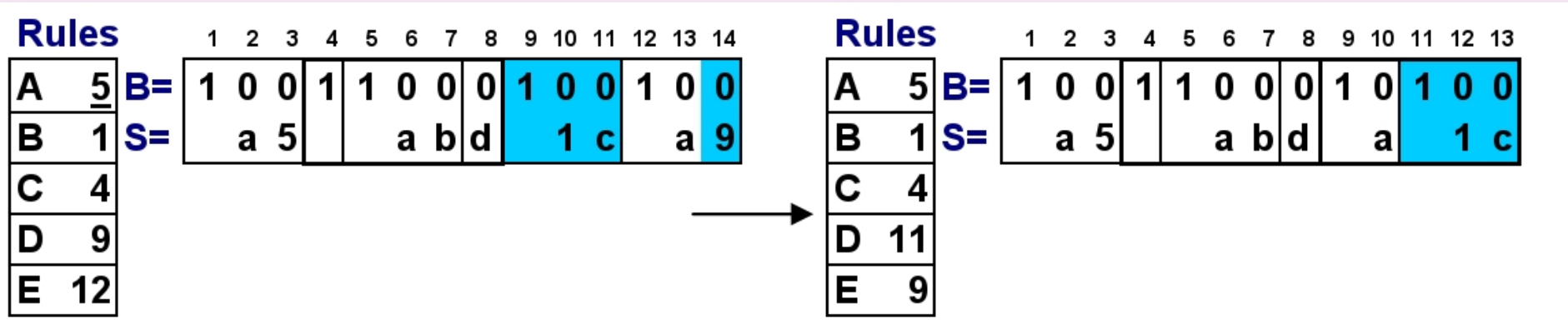
Rules

A	1
B	4
C	7
D	10
E	13

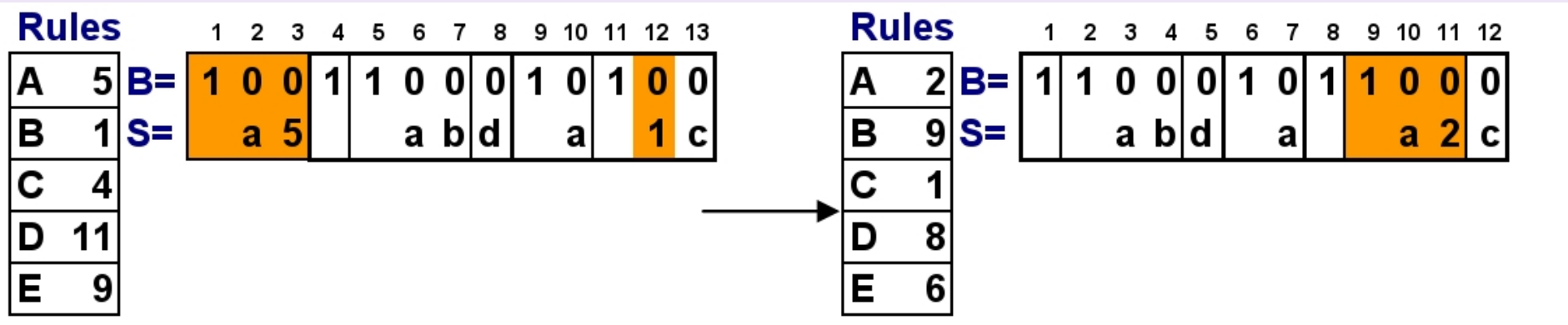
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
B=	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0
S=		a	b		a	1		1	d		4	c		a	10

Improving Dictionary Representation





Improving Dictionary Representation



Limiting the Dictionary Size

- We stop forming pairs when the dictionary no longer fits in RAM
- Compression ratio depends on relative dictionary size, not absolute
- Preliminary results: good compression with dictionary as small as 5% of text

Outline

1. Introduction
2. Compressing with Re-Pair
   - Exact Version
   - Approximate Version
3. Dictionary Compression
4. Local Decompression
5. Theoretical Analysis
6. Experimental Results
7. Conclusions and Future Work

Local Decompression

Adding structures

We add a bitmap BR of length n which marks every begining of phrase.

T	A	C	D	A	C	D	D	E	C	D	E
BR	1	0	0	1	0	0	1	1	1	0	1
C	G	G	D	E	F	E					

BR has to support rank and select operations

- $\texttt{rank}_{\texttt{BR}}(i)$: number of 1's in $\texttt{BR}[1..i]$
- $\texttt{select}_{\texttt{BR}}(i)$: position of the i-th 1 in BR

Decompressing

Using BR

Using BR we can decompress any substring expanding the optimal number of symbols.

T	A	C	D	A	C	D	D	E	C	D	E
BR	1	0	0	1	0	0	1	1	1	0	1
C	G	G	D	E	F	E					

Example

For extracting $T[2..5]$ we expand:

$$C[\mathtt{rank}_{\mathtt{BR}}(2)..\mathtt{rank}_{\mathtt{BR}}(5)]$$

Squeezing BR

- If the text is compressible, BR is sparse
- It can be compressed to H_0
- We implemented two compressed structures

Outline

1. Introduction
2. Compressing with Re-Pair
   - Exact Version
   - Approximate Version
3. Dictionary Compression
4. Local Decompression
5. Theoretical Analysis
6. Experimental Results
7. Conclusions and Future Work

Theoretical Analysis

Theorem (KM99)

Let $y_1 \dots y_t$ denote a parsing of the string $T[1, n]$ over an alphabet of size σ, such that each phrase y_i appears at most b times. For any $k \geq 0$ we have

$$t \log t \leq nH_k(T) + t \log(n/t) + t \log b + \Theta(t(1 + k \log \sigma))$$

Observation

After applying Re-Pair, every pair of consecutive symbols is unique. After replacing all the pairs that occur more than b times, every pair of consecutive symbols appears at most b times.

Theoretical Analysis

Theorem

Let $T[1,n]$ be a text over an alphabet of size σ and having k-th order (classical or empirical) entropy $H_k(T)$. Then, compression algorithm Re-Pair achieves a representation using at most $2nH_k(T) + o(n \log \sigma)$ bits for any $k = o(\log_\sigma n)$ (which implies $\log \sigma = o(\log n)$ unless $k = 0$).

Theoretical Analysis

The proof

- The size of the compressed sequence can be bounded using [KM99]
- We need to prove that every pair expands differently
- We prove that the dictionary does not grow too much

Outline

1. Introduction
2. Compressing with Re-Pair
   - Exact Version
   - Approximate Version
3. Dictionary Compression
4. Local Decompression
5. Theoretical Analysis
6. Experimental Results
7. Conclusions and Future Work

Compression Ratio (as %)

File	Repair	gzip -1	gzip -9	bzip2 -1	bzip2 -9
DNA	34.37	32.51	27.02	26.57	25.95
ENGLISH	29.06	44.99	37.64	32.83	28.07
PROTEINS	51.18	48.86	46.51	45.74	44.80
SOURCES	25.71	28.01	22.38	21.30	18.67
XML	13.93	21.19	17.12	14.37	11.36

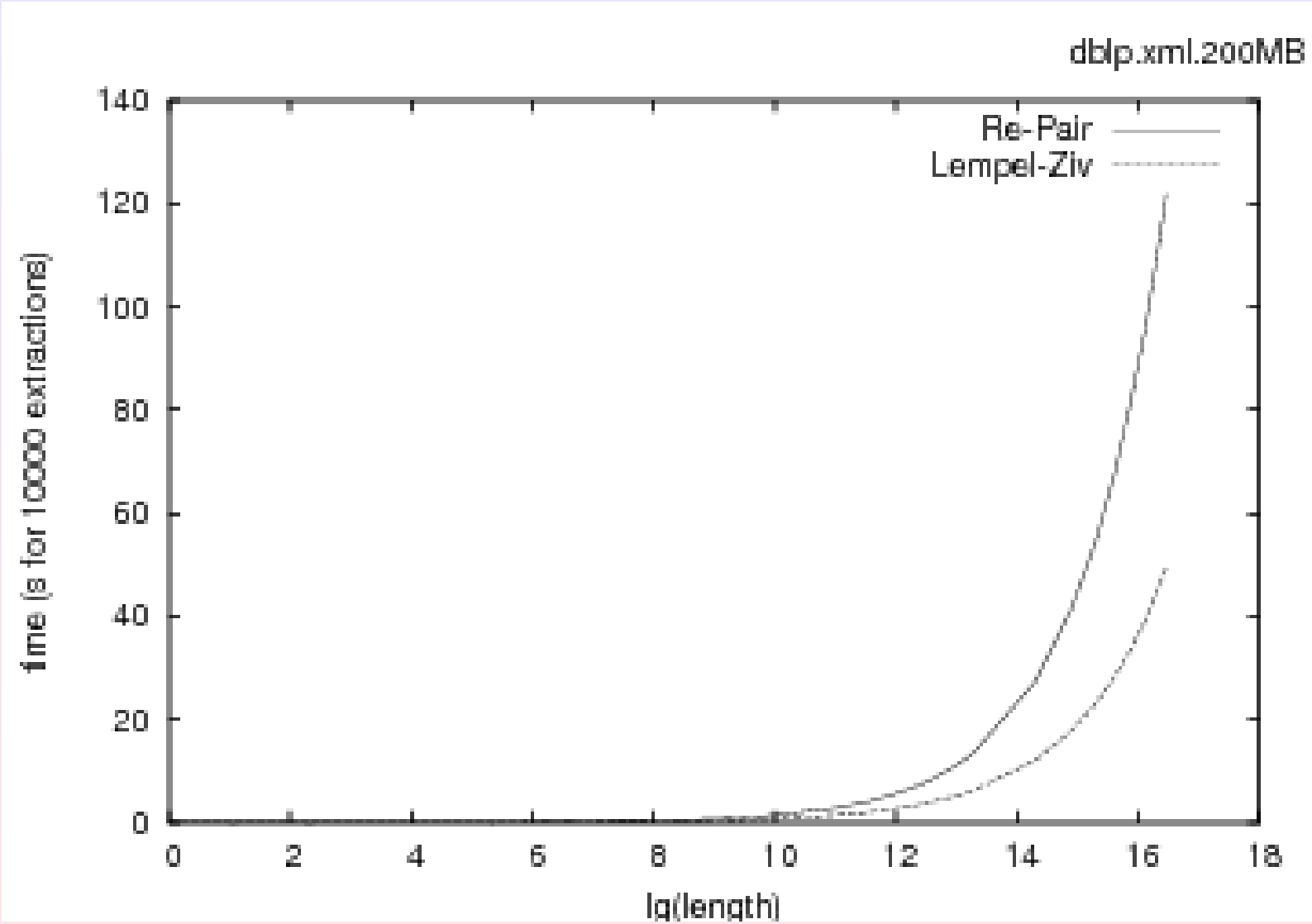
Approximate Version (as %)

File	Repair	appRepair
DNA	34.37	36.22
ENGLISH	29.06	34.30
PROTEINS	51.18	53.53
SOURCES	25.71	31.21
XML	13.93	17.10

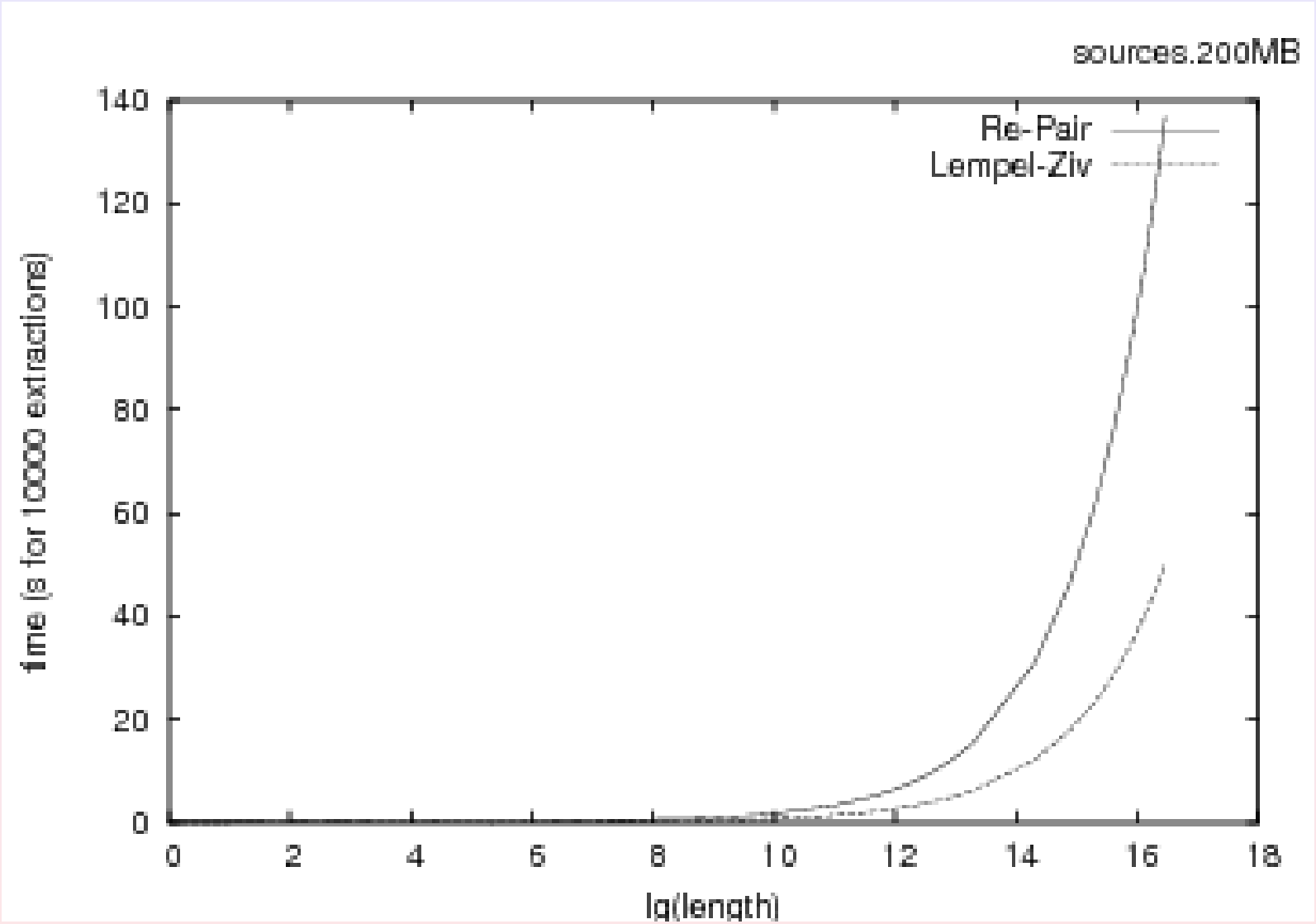
Compressing BR (sizes in bytes)

File	|Br|	|imp1|	|imp2|
dblp.xml.200MB	27525152	12527918	11043041
dna.200MB	27525152	18325588	17848550
english.200MB	27525152	15473324	15226928
pitches.50MB	6881312	4596241	4797069
proteins.200MB	27525152	17605940	18759922
sources.200MB	27525152	14809692	14209888

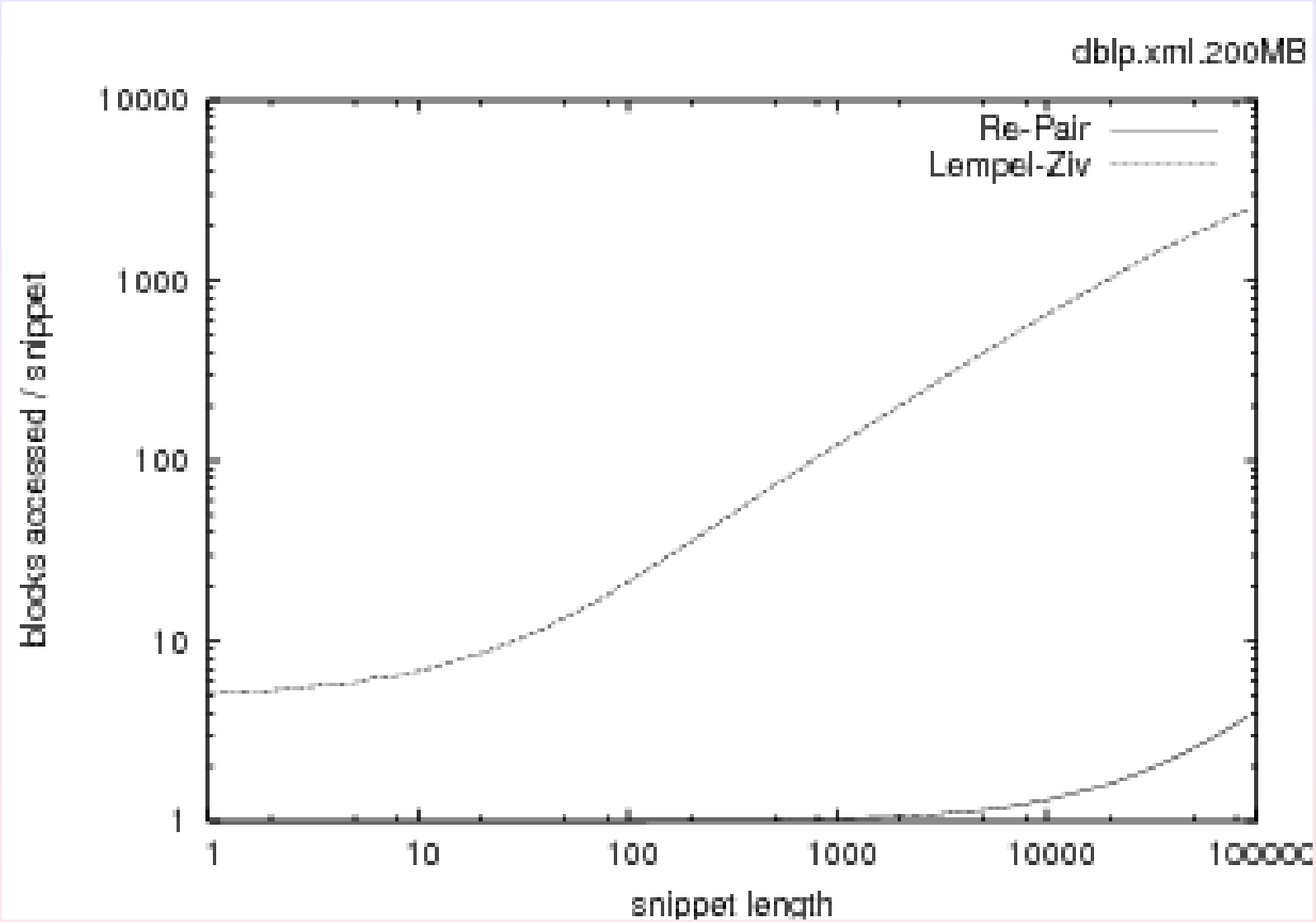
Decompression speed



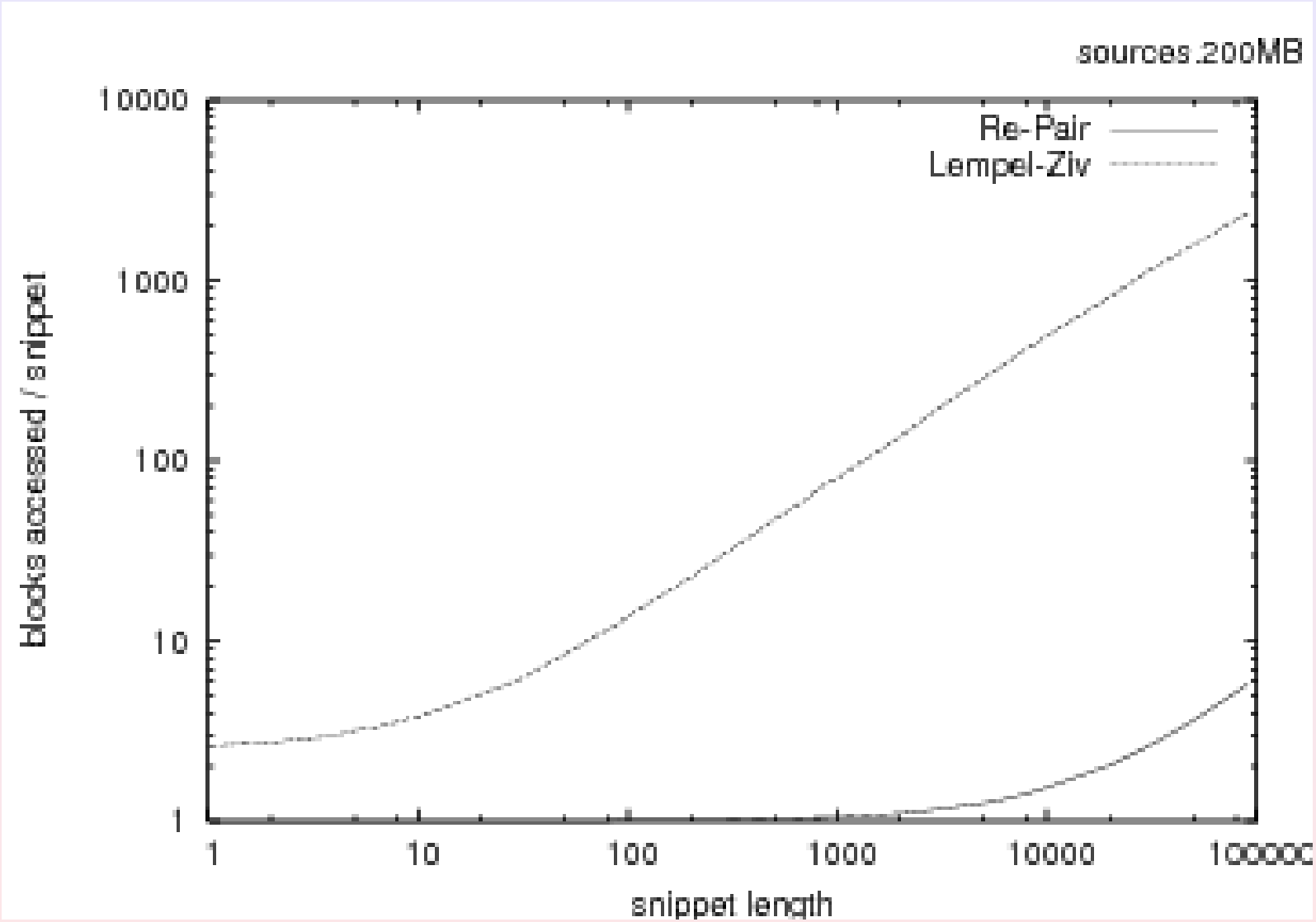
Decompression speed



Disk Accesses for Decompression



Disk Accesses for Decompression



Outline

1. Introduction
2. Compressing with Re-Pair
   - Exact Version
   - Approximate Version
3. Dictionary Compression
4. Local Decompression
5. Theoretical Analysis
6. Experimental Results
7. Conclusions and Future Work

Conclusions

- Re-Pair achieves good compression ratio
- Very fast decompression, especially on disk
- Can be implemented on secondary storage
- Allows compression ratio/speed tradeoff
- Achieves high-order entropy

Future Work

- Analyze the approximate version
- k variable during compression
- Decompression speed with limited dictionary

Questions?