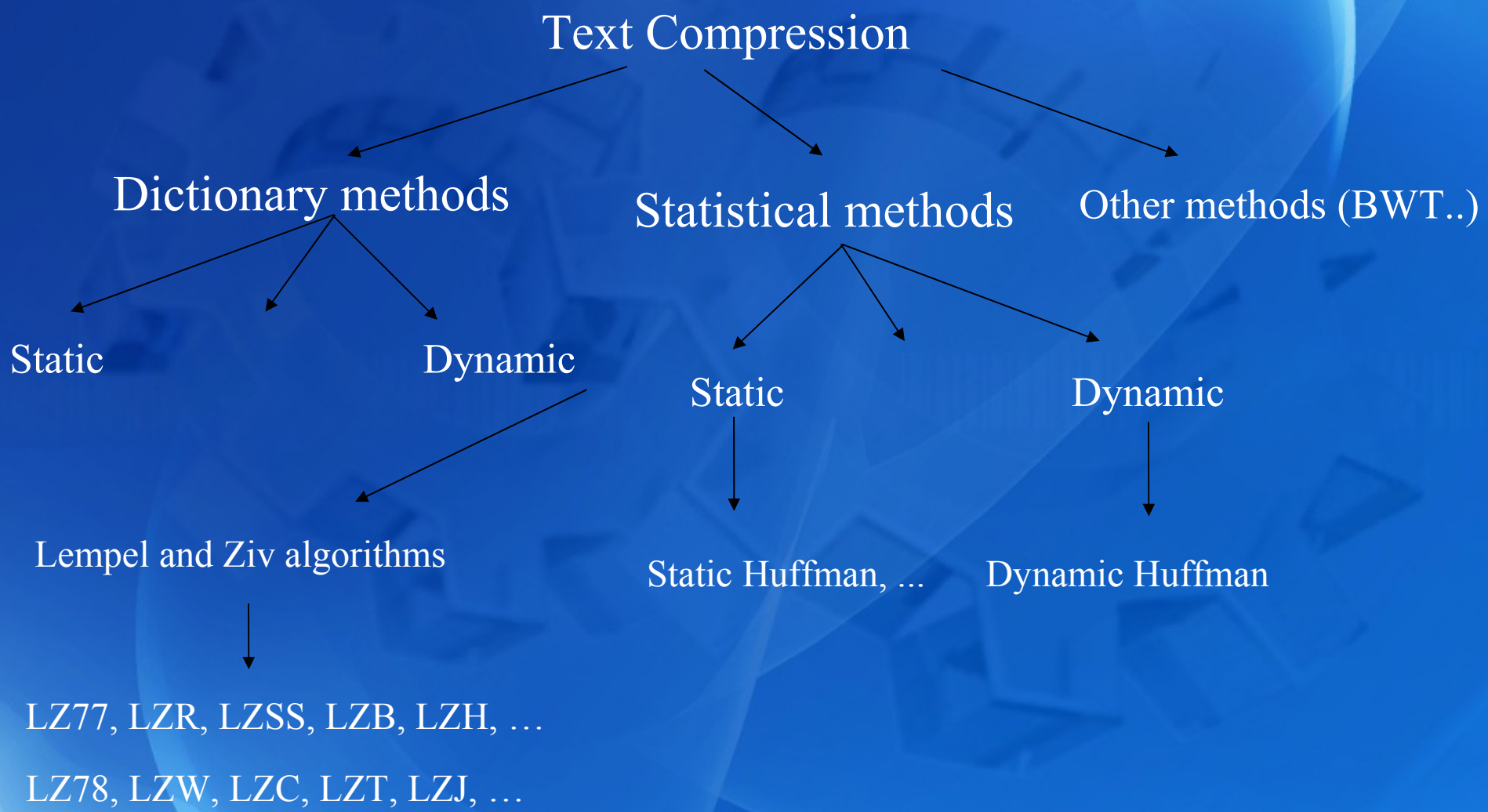
M. Crochemore, L. Giambruno, A. Langiu,
F. Mignosi, A. Restivo

Dictionary-Symbolwise Flexible Parsing

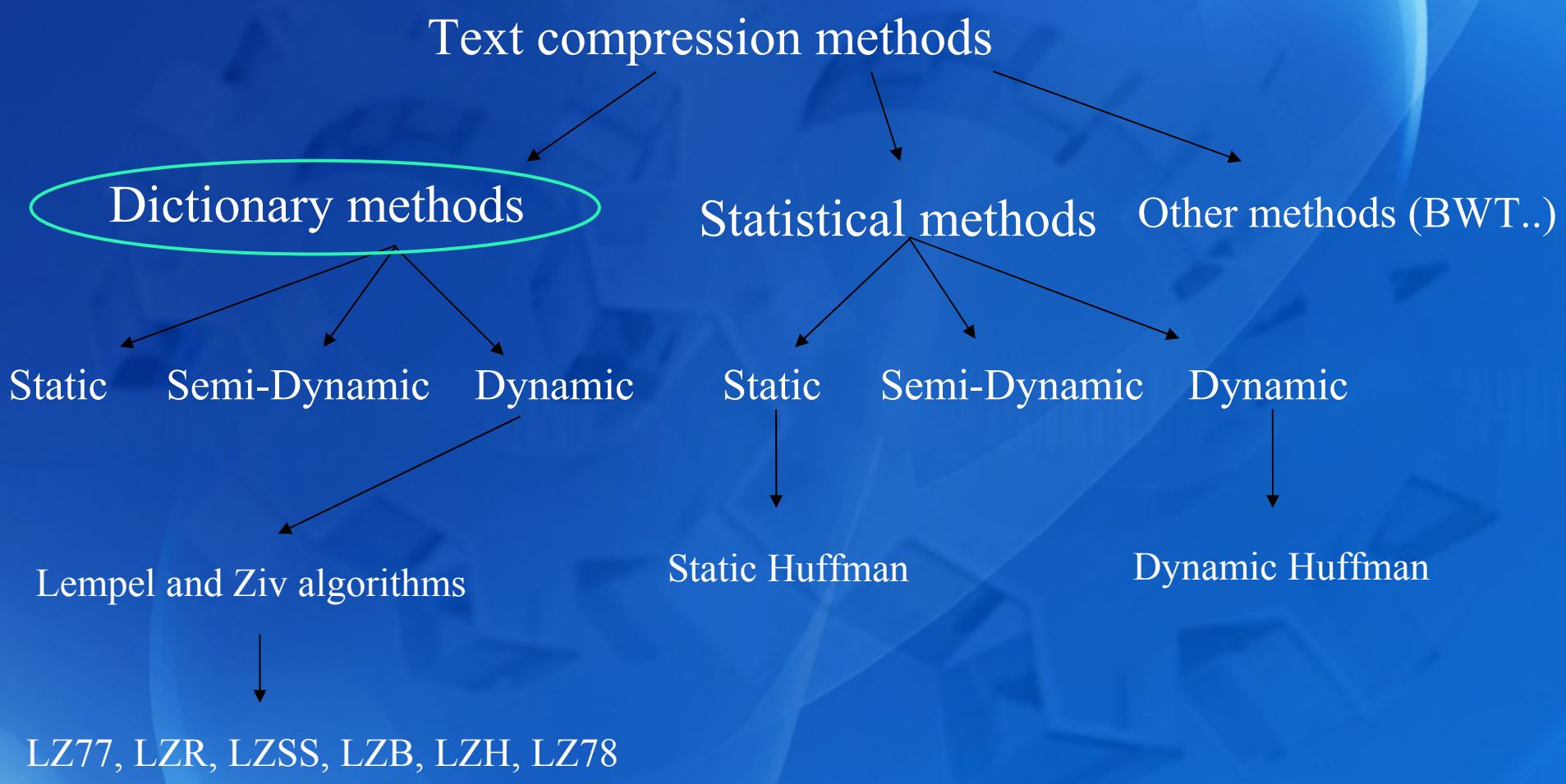
Plan of The Talk

- What are Dictionary-Symbolwise methods (DS)?
- Why DS?
- A bit of history
- Technical contribution
- Conclusions and problems

What are DS?


Text Compression
Dictionary methods
Statistical methods
Other methods (BWT..)
Static
Dynamic
Static
Dynamic
Lempel and Ziv algorithms
Static Huffman, ...
Dynamic Huffman
LZ77, LZR, LZSS, LZB, LZH, …
LZ78, LZW, LZC, LZT, LZJ, …

What are DS - Dictionary methods



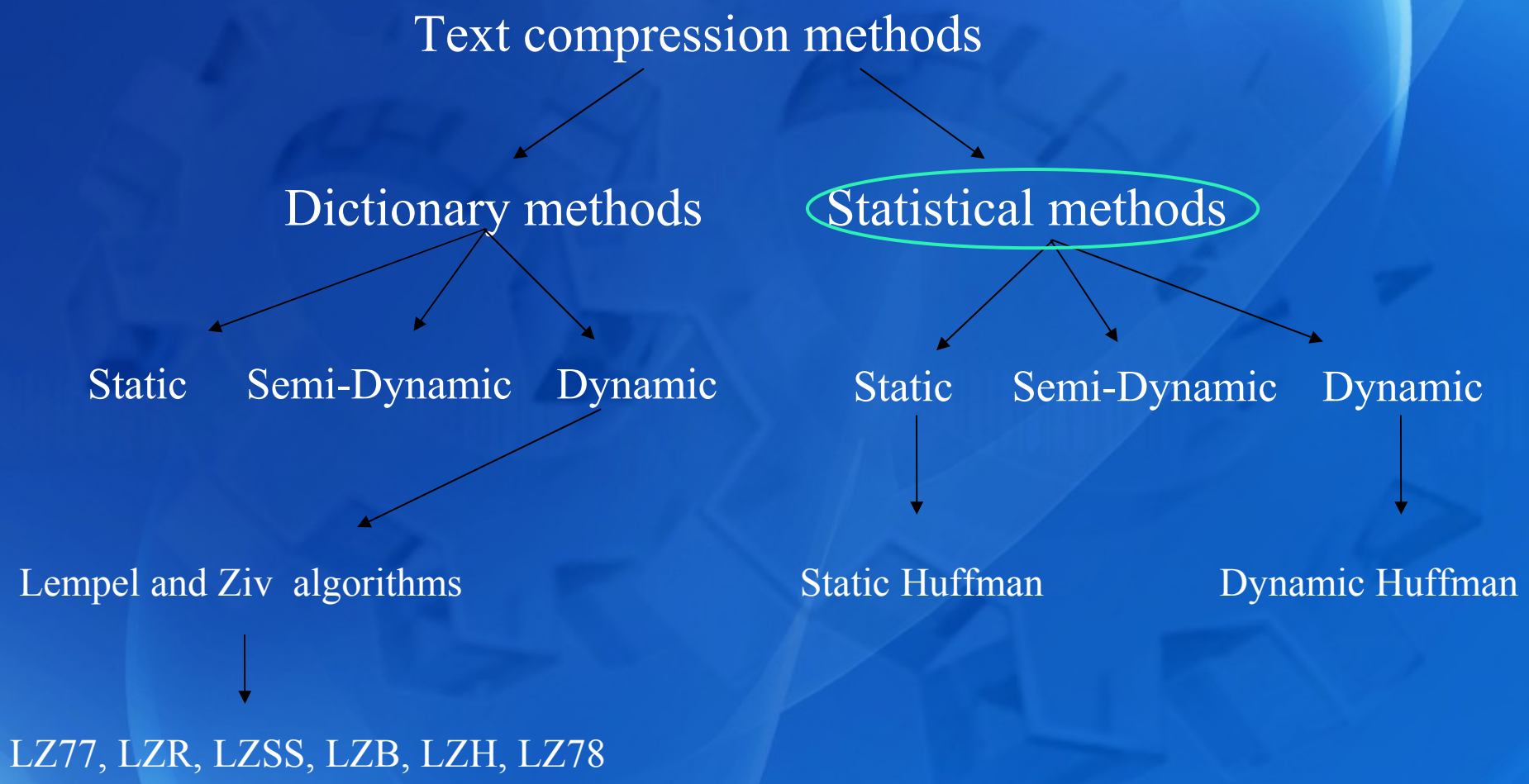
What are DS - Dictionary Methods

A *dictionary compression algorithm* replaces phrases with a code of a dictionary pointer (an index into a dictionary)

It can be fully specified by

(1) the dictionary description
(2) the encoding of dictionary pointers
(3) the parsing method, i.e. the algorithm that splits the text in dictionary phrases

What are DS – Symbolwise Methods



What are DS – Symbolwise Methods

- Also called statistical;
- Symbolwise methods replace letters by a code that can depend also on the context;
- Most frequent letters should have smaller code. Often it is used an arithmetic code;
- Classical example: Huffman;
- Can achieve great (best) compression ratio but in this case slow both in compression and in decompression. PPM, PPM*

What are DS?

The result of mixing the two methods.

It could be practical to make a free mixture of dictionary pointers and symbolwise codes.

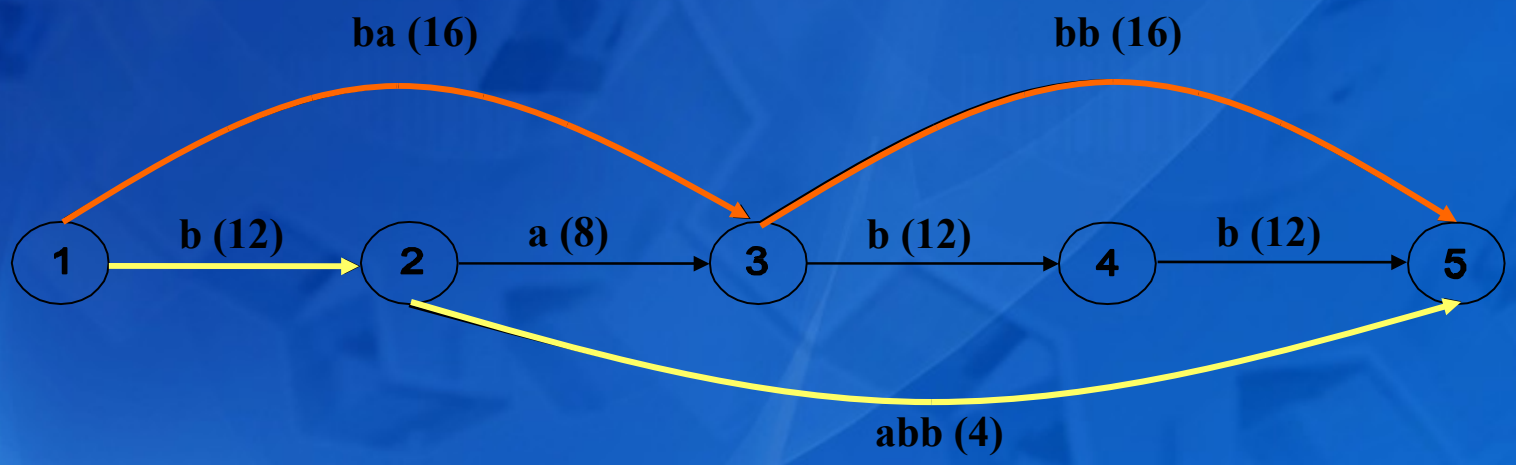
Is it possible to make an optimal (cost) choice of this mixture?

If yes, is it possible to get it (a parse of the text) in linear time?

Example: with a dictionary (static)

String: babb

m	*C(m)*
a	8
b	12
ba	16
bb	16
abb	4



Cost of the Greedy solution: 32

Cost of the optimal solution: 16

Example: symbolwise (static)

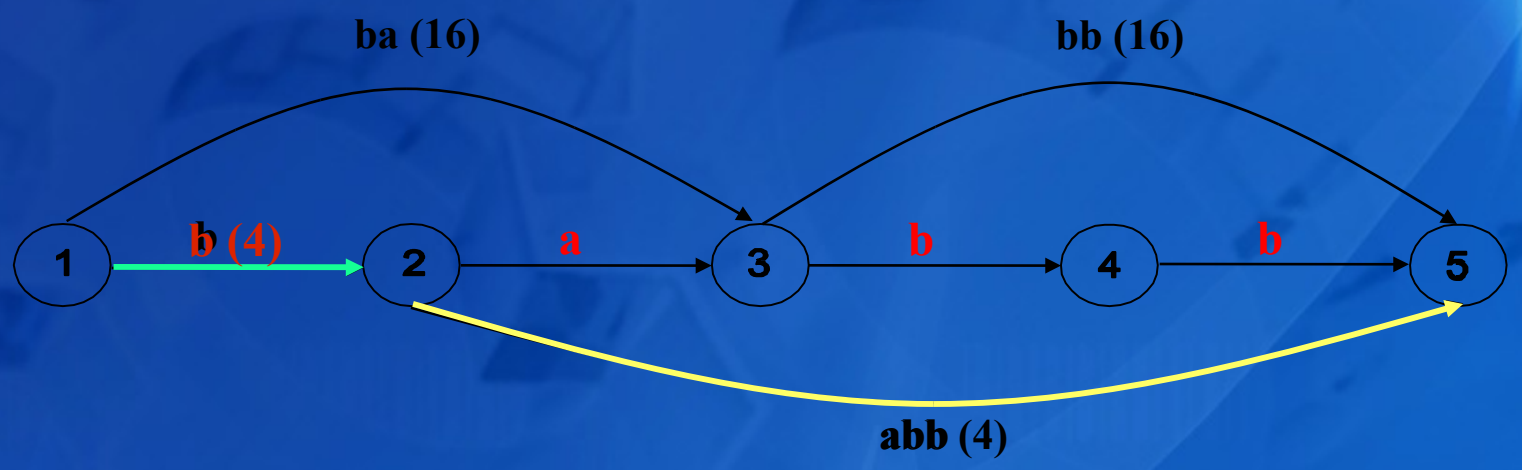
String: babb

Symbol	*Cost of the code*
a	8
b	4

Cost of the (optimal) solution: 20

Example: dictionary-symbolwise

Why not to use both?



m	*C(m)*
a	8
b	12
ba	16
bb	16
abb	4

Symbol	*Cost of the code*
a	8
b	4

Cost of the optimal solution: 10 = 8 + 2

What are DS?

- DS algorithms can be fully specified by (1) the dictionary description (2) the pointers encoding (3) the symbolwise algorithm (4) the encoding of the flag information (5) the (generalized) parsing method
- Gzip and zip and cabarc belong to this class. They are fast in decompression because the symblowise algorithm is fast (Huffman)

Why DS?

AND THE WINNER IS (decompression time) ..... cabarc

See Large Text Compression Benchmark http://mattmahoney.net/text/text.html

- Coupling an heavy symbolwise with a dictionary method one can reduce decompression time without degrading compression ratio

Why DS?

Coupling an heavy symbolwise with a dictionary method one can reduce decompression time without degrading compression ratio

DS are used in many relevant commercial compressors. PROBABLY

A BIT OF HISTORY

DS are used BUT the theory is not developed

(G. Della Penna and A. Langiu and F. Mignosi and A. Ulisse, "Optimal Parsing in Dictionary-Symbolwise Data Compression Schemes",http://www.di.univaq.it/~mignosi/ulicompressor.php}, 2006 and in this contribution.)

2 Research Lines are related

- Optimal Parsing and Graphs
- Flexible Parsing

A BIT OF HISTORY: graphs

- Initial state plus a vertex for each symbol;
- Arcs outgoing each vertex. One to next symbol (labeled by the symbolwise cost of the symbol+flag) and the others the dictionary arcs (labeled by the corresponding cost+flag);
- OPTIMAL PARSING <<>> MINIMAL PATHS
- It is a DAG; therefore the minimal path can be found in O(|E|)

A BIT OF HISTORY: graphs

ABOVE GRAPH IS NOT ALWAYS WELL DEFINED!

- O(|E|) can be quadratic. Choose a subraph (how?)

- Schuegraf and Heaps (1974) for static dictionaries. Considered for static dictionaries up to 2006 by many. Zip-Deflate64 ?, rzm (2008)?
- Ferragina Nitto Venturini (SODA 2009) LZ77 variable logaritmic costs, subgraph O(nlog(n)).

A BIT OF HISTORY: Flexible Parsing

- Hartman and Rodeh 1984. Greedy with one step lookahead. Static Dict, Constant Cost of Pointers, Prefix closed Dictionaries. Optimal
- Horspool 1995. Strange nice Dictionary LZ78-alike. Optimal without knowing. Constant Cost
- Kim Kim explicitely dynamic 1998. Const Cost. Strange Dictionary LZ77-alike for which Greedy=Flexible.

OUR CONTRIBUTION

We follow above research lines in the DS case.

- We discover a subgraph that "preserves" minimal paths. OPTIMAL parsing
- For building our subgraph we used also a procedure used by Mathias and Shainalp.
- Prefix closed Dict. BUT costs are not constant

TECHNICAL CONTRIBUTION

HOW TO BUILD OUR SUBGRAPH?

- We add the symbolwise arcs (linear)
- We look for c-supermaximal arcs (not “contained” in other c-arcs) where the cost of the arc is <=c.
- For each such arc (x,y) we add to the subgraph (x,y-1), (x,y-2), … until y-k is either a previous ending point of another supermaximal arc or it is x+1 (prefix closed)

TECHNICAL CONTRIBUTION

- IT IS A SIMPLE SUBGRAPH
- To find the c-supermaximal, our procedure is very similar to one of Mathias and Shainalp for the flexible parsing.
- Linear for LZ78-alike, O(nlog(n)) for LZ77-alike.

Conclusions

- We reached the compression ratio of some main commercial programs (LZ77).
- Experiments shows that using Huffman as Symbolwise, LZ78 and Optimal parsing there is a gain of around 5% wrt no Huffman. (20 MB texts). Higer gain for smaller text. For LZ77 there is a 2.5% gain

Conclusions

- The DS theory seems to be the good one for many methods.
- Thanks to our results it is possible to optimize the trade-off between some main parameters such as decompression time, compression ratio, compression time.

Open Problems

- Theorically LZ78 is better on memoryless sources than LZ77. Experimental results say that when Optimal parsing is in use it happens the opposit. Prove this fact.
- When higer order symbolwise is in use the distance between the compression ratios of LZ77 and LZ78 is smaller. Prove it.

THANKS