

On Dictionary-Symbolwise Data Compression

G. Della Penna¹, F. Mignosi¹, A. Langiu², A. Ulisse¹

¹ Dipartimento di Informatica, Università dell'Aquila, Italy

² Dipartimento di Matematica e Applicazioni, Università di Palermo, Italy

1 Introduction

To our best knowledge, we are the first to use the name "dictionary-symbolwise". Many algorithms or programs that, following our definition, are dictionary-symbolwise have been classified up to now just as "dictionary" algorithms or programs. In some sense, and this is a main contribution of this paper, we are splitting in two parts what was, up to now, considered the class of dictionary algorithms. Indeed, previous classification was unsatisfactory for several reasons and our more accurate subdivision allowed us to improve previous results and to obtain new ones. Famous viable programs, such as *gzip* and *zip* are dictionary-symbolwise, according to us.

In the scientific literature, there are many data compression algorithms that can be mixed in a sort of "cocktail" to improve compression rates. In particular, there are two famous compression methods that can work together: the *symbolwise* and the *dictionary* encoding, which are sometimes referred to as *statistical* encoding and *parsing* (or macro) encoding, respectively. In [2] it is possible to find a survey on these schemes and a description of deep relationships among them (see also [22]).

For us, a data compression algorithm that uses *both* a dictionary and a symbolwise encoding belongs to the class of *dictionary-symbolwise* compressors. Bell, in his 1986 paper ([1]) implements a suggestion made earlier by Storer and Szymanski in 1982 (see [21]), i.e. to use a free mixture of dictionary pointers and literal characters in the Lempel-Ziv algorithms, literals only being used when a dictionary pointer takes up more space than the characters it codes. Bell's implementation of this scheme is called LZSS, and adds an extra bit to each pointer or character to distinguish between them. LZSS uses no symbolwise compression but, since the identity can be viewed as a particular symbolwise compressor, from a purely formal point of view we can say that LZSS is dictionary-symbolwise and that the whole class of dictionary-symbolwise compressors conceptually derives from LZSS.

Many papers deal with *optimal parsing* in dictionary compressions (see for instance [21, 19, 13, 2, 5, 14, 17, 16, 23, 15, 9]), that is a way of parsing the input string that minimizes the cost of the encoded output. A classic way for obtaining an optimal parsing is by reduction to a well-known graph theoretical problem. This holds true also for dictionary-symbolwise compression, if we consider a natural extension of the optimality notion to such kind of algorithms.

This approach, however, is not recommended in [20] because it is too time consuming. In [16] a pruning mechanism is described, that allows a more efficient computation and that still enables the evaluation of an optimal solution. The pruning process may be applied in all cases for which the cost function satisfies the triangle inequality.

In this paper we show a novel method that allows to obtain an optimal parsing in linear time of any input under a natural hypothesis on the encoding algorithm and on the cost function. In particular, the constraint on the cost function is *weaker* than the triangle inequality. Indeed, in LZ78 and similar schemes we just require the cost of dictionary pointers to be constant. This hypothesis is natural in this kind of schemes and it has been used in the main recent optimal-parsing results (cf. for instance [5, 19]). In LZ77-derived scheme, using additional arcs, our method obtain an optimal parsing also when the cost of dictionary pointers depends logarithmically on the dictionary phrase size, and it is linear for all practical purposes. This is the very first time that such a kind of result is obtained even for some subclass of LZ77-derived algorithms.

Our result for the LZ78 case can also be seen as a generalization to the dictionary-symbolwise case of the result presented in [5] and also of the main result obtained in [19]. Concerning LZ77-derived algorithms, it improves the non-greedy parsing considered in [10] and used by *gzip*. The proposed

technique works for any kind of dictionary, static or dynamically changing, for any pointer-coding function and for any symbolwise compression. This includes all cases considered in [16].

More in detail, our method shows how to build, in the case of LZ78-derived algorithms, in linear time a subgraph with linear size of the (generalized) classical graph model considered in [16], without using any pruning mechanism. Moreover, in the case of LZ77 compression schemes, we show how to exploit CDAWGs (Compact Directed Acyclic Word Graphs, c.f. [3, 7, 11, 12]) to build such parsing graph. To our best knowledge, this is the first time that CDAWGs are in LZ77 algorithms, instead of classical structures like DAWGs (non compacted Directed Acyclic Word Graphs, cf. ([6]) or suffix trees ([18]), hashing ([4, 8]). Thanks to CDAWGs, under some additional hypothesis, it is also possible to build the graph and achieve an optimal parsing in an *online* linear manner.

References

1. T. C. Bell. Better opm/l text compression. *IEEE Trans. Comm.*, COM-34(12):1176–1182, 1986.
2. Timothy C. Bell and Ian H. Witten. The relationship between greedy parsing and symbolwise text compression. *J. ACM*, 41(4):708–724, 1994.
3. Anselm Blumer, J. Blumer, Andrzej Ehrenfeucht, David Haussler, and Ross M. McConnell. Building the minimal dfa for the set of all subwords of a word on-line in linear time. In Jan Paredaens, editor, *ICALP*, volume 172 of *Lecture Notes in Computer Science*, pages 109–118. Springer, 1984.
4. Richard P. Brent. A linear algorithm for data compression. *Australian Computer Journal*, 19(2):64–68, 1987.
5. Martin Cohn and Roger Khazan. Parsing with prefix and suffix dictionaries. In *Data Compression Conference*, pages 180–189, 1996.
6. Maxime Crochemore. Transducers and repetitions. *Theoretical Computer Science*, 45(1):63–86, 1986.
7. Maxime Crochemore and Renaud Verin. Direct construction of compact directed acyclic word graphs. In Alberto Apostolico and Jotun Hein, editors, *CPM*, volume 1264 of *Lecture Notes in Computer Science*, pages 116–129. Springer, 1997.
8. Gzip’s Home Page: <http://www.gzip.org>.
9. Alan Hartman and Michael Rodeh. *Optimal parsing of strings*, pages 155–167. Springer - Verlag, 1985.
10. R. Nigel Horspool. The effect of non-greedy parsing in ziv-lempel compression methods. In *Data Compression Conference*, pages 302–311, 1995.
11. Shunsuke Inenaga, Hiromasa Hoshino, Ayumi Shinohara, Masayuki Takeda, Setsuo Arikawa, Giancarlo Mauri, and Giulio Pavesi. On-line construction of compact directed acyclic word graphs. *Discrete Applied Mathematics*, 146(2):156–179, 2005.
12. Shunsuke Inenaga, Ayumi Shinohara, Masayuki Takeda, and Setsuo Arikawa. Compact directed acyclic word graphs for a sliding window. In *SPIRE*, pages 310–324, 2002.
13. Jyrki Katajainen and Timo Raita. An approximation algorithm for space-optimal encoding of a text. *Comput. J.*, 32(3):228–237, 1989.
14. Jyrki Katajainen and Timo Raita. An analysis of the longest match and the greedy heuristics in text encoding. *J. ACM*, 39(2):281–294, 1992.
15. Tae Young Kim and Taejeong Kim. On-line optimal parsing in dictionary-based coding adaptive. *Electronic Letters*, 34(11):1071–1072, 1998.
16. Shmuel T. Klein. Efficient optimal recompression. *Comput. J.*, 40(2/3):117–126, 1997.
17. Yossi Matias, Nasir Rajpoot, and Sileyman Cenk Sahinalp. The effect of flexible parsing for dynamic dictionary-based data compression. *ACM Journal of Experimental Algorithms*, 6:10, 2001.
18. Joong Chae Na, Alberto Apostolico, Costas S. Iliopoulos, and Kunsoo Park. Truncated suffix trees and their application to data compression. *Theor. Comput. Sci.*, 1-3(304):87–101, 2003.
19. Yossi Matias and Sileyman Cenk Sahinalp. On the optimality of parsing in dynamic dictionary based data compression. In *SODA*, pages 943–944, 1999.
20. Ernst J. Schuegraf and H. S. Heaps. A comparison of algorithms for data base compression by use of fragments as language elements. *Information Storage and Retrieval*, 10(9-10):309–319, 1974.
21. James A. Storer and Thomas G. Szymanski. Data compression via textural substitution. *J. ACM*, 29(4):928–951, 1982.
22. John G. Cleary Timothy C. Bell and Ian H. Witten. *Text compression*. Prentice Hall, 1990.
23. Robert A. Wagner. Common phrases and minimum-space text storage. *Commun. ACM*, 16(3):148–152, 1973.