Suffix Arrays: What Are They Good For?

Simon J. Puglisi
Dept. of Computing
Curtin University of Technology
Perth, Australia
Email: puglissj@computing.edu.au

William F. Smyth
McMaster University, Hamilton, Canada
Curtin University of Technology, Perth, Australia
Email: smyth@computing.edu.au

Andrew Turpin
Dept. of Computer Science & IT
RMIT University
Melbourne, Australia
Email: aht@cs.rmit.edu.au

Recently the theoretical community has displayed a flurry of interest in suffix arrays, and compressed suffix arrays. New, asymptotically optimal algorithms for construction, search, and compression of suffix arrays have been proposed. In this talk we will present our investigations into the practicalities of these latest developments. In particular, we investigate whether suffix arrays can indeed replace inverted files, as suggested in recent literature on suffix arrays.

Background

In 1990 Manber & Myers proposed suffix arrays as a space-saving alternative to suffix trees and described the first algorithms for suffix array construction and use (Manber & Myers 1990, Manber & Myers 1993). It has since been shown that any problem whose solution can be computed using suffix trees is solvable with the same asymptotic complexity using suffix arrays (Abouelhoda, Kurtz & Ohlebusch 2004). In addition, suffix arrays use much less memory than suffix trees, even less when they are compressed (Ferragina & Manzini 2000, Sadakane 2002, Grossi, Vitter & Gupta 2004, Puglisi, Turpin & Smyth 2005*a*, Mäkinen & Navarro 2005).

It has recently been shown that given an n character text T and its corresponding suffix array S, with some preprocessing and auxiliary information, it is possible to search for an arbitrary m character pattern P in T using only $O(m)$ time (Sim, Kim, Park & Park 2003). This is superior to non-index based string matching algorithms like that of Knuth, Morris & Pratt (1977) and Boyer & Moore (1977) which are linear in both the pattern and text length, requiring $O(m + n)$ time to find P in T. In conjunction with these time-efficient searching algorithms, time-efficient construction algorithms have also been developed that require only $O(n)$ time to construct the suffix array on an n character text (Puglisi, Turpin & Smyth 2005*b*).

Subsequent research on compressed suffix arrays (Sadakane 2000, Mäkinen 2000, Grossi & Vitter n.d.) and similar structures has revealed that *self-indexing* structures are possible, which can search for and report matches without the need for the original text to be stored (Mäkinen & Navarro 2004, Ferragina & Manzini 2000, Ferragina & Manzini 2001, Navarro 2004, Grossi et al. 2004). These structures typically require about 30% of the space of the text, and so double as a compression scheme as the original text can be discarded. Search times remain linear in the length of the pattern (assuming a fixed alphabet, such as ASCII).

While a great deal of effort has been expended in making suffix arrays smaller, there is still a fundamental problem with their scalability. When searching for a pattern P of length m, one must perform m non-sequential accesses into the suffix array, and m non-sequential access into the text. If the suffix array is on disk, this equates to $2m$ seek operations, which, for anything but small patterns (of the order of 5 characters), limits the technology to a small number of simultaneous users, or small texts that fit in RAM. Even the compressed, self-indexing suffix array of Grossi et al. (2004), which does not require access to the text, requires $O(m + \log n)$ seeks into the structure itself.

Because of the non-sequential access patterns exhibited by current suffix array algorithms, all papers experimenting with such algorithms assume that their structures can fit in memory. This seems to contradict bold claims that suffix arrays are an important technology for searching the World Wide Web, and even large genomic databases (Sadakane & Shibuya 2001, Grossi et al. 2004).

The inverted file, on the other hand, is a data structure that has been adopted by the Web search engine community, and handles data on external storage (Witten, Moffat & Bell 1999). Inverted files have been specifically engineered to scale well, and to minimise the number of expensive disk operations required to find a pattern in a text (Zobel & Moffat n.d.). However, the form of the pattern is restricted. With an inverted file, the form of the pattern must be set prior to index construction. Typically

a word is chosen as the unit of indexing, restricting pattern search to words, prefixes of words, or combinations of words (phrases).

In this talk we will report on experiments with inverted files in direct competition to suffix arrays: that is, all data is in RAM, and arbitrary patterns are the target of the search.

Acknowledgments This work has been supported in part by grants from the Australian Research Council (Turpin) and the Natural Sciences and Engineering Research Council of Canada (Smyth).

References

Abouelhoda, M. I., Kurtz, S. & Ohlebusch, E. (2004), 'Replacing suffix trees with enhanced suffix arrays', *Journ. Discrete Algorithms* **2**, 53–86.

Boyer, R. S. & Moore, J. S. (1977), 'A fast string searching algorithm', *Communications of the ACM* **20**(10), 762–772.

Ferragina, P. & Manzini, G. (2000), Opportunistic data structures with applications, *in* 'Proceedings of the 41st IEEE Symposium on Foundations of Computer Science (FOCS 00)', IEEE Computer Society, Redondo Beach, CA, pp. 390–398.

Ferragina, P. & Manzini, G. (2001), An experimental study of an opportunistic index, *in* 'SODA '01: Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms', Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, pp. 269–278.

Grossi, R. & Vitter, J. S. (n.d.), 'Compressed suffix arrays and suffix trees with applications to text indexing and string matching', *SIAM Journal on Computing* . To appear. Available from `http://www.cs.duke.edu/~jsv/Papers/catalog/node87.html`.

Grossi, R., Vitter, J. S. & Gupta, A. (2004), When indexing equals compression: Experiments with compressing suffix arrays and applications, *in* 'SODA '04: Proceedings of the fifteenth annual ACM-SIAM Symposium on Discrete algorithms', SIAM, New Orleans, Louisianna, USA, pp. 636–645.

Knuth, D. E., Morris, J. H. & Pratt, V. R. (1977), 'Fast pattern matching in strings', *SIAM Journal on Computing* **6**(2), 323–350.

Mäkinen, V. (2000), Compact suffix array, *in* 'Combinatorial Pattern Matching', Vol. LNCS 1848, pp. 305–319.

Mäkinen, V. & Navarro, G. (2004), Compressed compact suffix arrays, *in* 'Combinatorial Pattern Matching: 15th Annual Symposium, CPM 2004', Vol. LNCS 3109, Springer-Verlag GmbH, pp. 420–433.

Mäkinen, V. & Navarro, G. (2005), 'Succinct suffix arrays based on run-length encoding', *Nordic Journal of Computing* **12**(2), 40–66.

Manber, U. & Myers, G. (1990), Suffix arrays: a new method for on-line string searches, *in* 'SODA '90: Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms', Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, pp. 319–327.

Manber, U. & Myers, G. W. (1993), 'Suffix arrays: a new model for on-line string searches', *SIAM Journal of Computing* **22**(5), 935–948.

Navarro, G. (2004), 'Indexing text using the Ziv-Lempel trie', *Journal of Discrete Algorithms* **2**(1), 87–114.

Puglisi, S. J., Turpin, A. H. & Smyth, W. F. (2005*a*), The performance of linear time suffix sorting algorithms, *in* M. Cohn & J. Storer, eds, 'Proceedings of the IEEE Data Compression Conference', IEEE Computer Society Press, Los Alamitos, CA, pp. 358–368.

Puglisi, S. J., Turpin, A. H. & Smyth, W. F. (2005*b*), A taxonomy of suffix array construction algorithms, *in* 'Proceedings of the Prague Stringology Conference', Czech Technical University, Prague, pp. 1–30.

Sadakane, K. (2000), Compressed text databases with efficient query algorithms based on the compressed suffix array, *in* 'Proceedings of ISSAC'00', Vol. LNCS 1969, pp. 410–421.

Sadakane, K. (2002), Succinct representations of lcp information and improvements in the compressed suffix arrays, *in* 'SODA '02: Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms', Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, pp. 225–232.

Sadakane, K. & Shibuya, T. (2001), 'Indexing huge genome sequences for solving various problems', *Genome Informatics* **12**, 175–183.

Sim, J. S., Kim, D. K., Park, H. & Park, K. (2003), Linear-time search in suffix arrays, *in* 'Proc. 14th Australian Workshop Combinatorial Alg. (AWOCA)', pp. 139–146.

Witten, I. H., Moffat, A. & Bell, T. C. (1999), *Managing Gigabytes: Compressing and Indexing Documents and Images*, second edn, Morgan Kaufmann Publishing, San Francisco.

Zobel, J. & Moffat, A. (n.d.), 'Inverted files for text search engines'. Submitted for publication.