# A New Compressed Suffix Tree Supporting Fast Search and Its Construction Algorithm Using Optimal Working Space*

Dong Kyue Kim[1] and Heejin Park[2,**]

[1] School of Electrical and Computer Engineering
Pusan National University, Busan 609-735, South Korea
[2] College of Information and Communications
Hanyang University, Seoul 133-791, South Korea
hjpark@hanyang.ac.kr

**Abstract.** The compressed suffix array and the compressed suffix tree for a given string $S$ are full-text index data structures occupying $O(n \log |\Sigma|)$ bits where $n$ is the length of $S$ and $\Sigma$ is the alphabet from which symbols of $S$ are drawn. When they were first introduced, they were constructed from suffix arrays and suffix trees, which implies they were not constructed in optimal $O(n \log |\Sigma|)$-bit working space. Recently, several methods were developed for constructing compressed suffix arrays and compressed suffix trees in optimal working space. By these methods, one can construct compressed suffix trees supporting the pattern search in $O(m'|\Sigma|)$ time where $m' = m \log^{\epsilon} n$, $m$ is the length of a pattern, and $\log^{\epsilon} n$ is the time to find the $i$th smallest suffix of $S$ from the compressed suffix array for any fixed $0 < \epsilon \leq 1$. However, compressed suffix trees supporting the pattern search in $O(m' \log |\Sigma|)$ time are not constructed by these methods.
In this paper, we present a new compressed suffix tree supporting $O(m' \log |\Sigma|)$-time pattern search and its construction algorithm using optimal working space. To obtain this result, we developed a new succinct representation of the suffix trees, which is different from the classic succinct representation of parentheses encoding of the suffix trees. Our succinct representation technique can be generally applicable to succinct representation of other search trees.

## 1 Introduction

A full-text index data structure for a text incorporates the indices for all the suffixes of the text. Two fundamental full-text index data structures are suffix trees [26, 31] and suffix arrays [10, 25], and many efficient algorithms have been developed for constructing suffix trees [6, 26, 31] and suffix arrays [18, 19, 22, 23]. They are used in numerous applications [14], which are exact string matching, computing matching statistics, finding maximal repeats, finding longest common substrings, and so on.

* This work was supported by Korea Research Foundation grant KRF-2003-03-D00343.

** Contact Author

A. Apostolico, M. Crochemore, and K. Park (Eds.): CPM 2005, LNCS 3537, pp. 33–44, 2005.

There have been efforts to develop a full-text index data structure that has the capabilities of both suffix trees and suffix arrays without requiring much space. The enhanced suffix array due to Abouelhoda et al. [1, 2] is such an index data structure. It consists of a `pos` array (suffix array), an `lcp` array, and a child table. The child table stores the parent-child relationship between nodes in a given suffix tree. Thus, on the enhanced suffix array, every algorithm developed either on suffix trees or suffix arrays can be run with a small and systematic modification. The only drawback of the enhanced suffix array is that the child table supports $O(m|\Sigma|)$-time pattern search where $m$ is the length of a pattern and $\Sigma$ is an alphabet. Recently, Kim et al. [20, 21] developed a new child table supporting $O(m \log |\Sigma|)$-time pattern search. They called the enhanced suffix array with the new child table linearized suffix tree.

Although many useful full-text index data structures are developed, their space consumption ($O(n \log n)$ bits for a string of length $n$) motivates researchers to develop more space efficient one. Compressed suffix arrays and compressed suffix trees are space efficient full-text index data structures that consume $O(n \log |\Sigma|)$ bits. Munro et al. [28] developed a succinct representation of a suffix tree topology under the name of space efficient suffix trees. Grossi and Vitter [12, 13] developed compressed suffix arrays, which takes $O(\log^{\epsilon} n)$ time to find the $i$th lexicographically smallest suffix in the compressed suffix array for any fixed $0 < \epsilon \leq 1$. Ferragina and Manzini [7, 8] suggested opportunistic data structures under the name of FM-index. Sadakane [30] modified the compressed suffix array so that it acts as a self-indexing data structure. The compressed suffix array and the FM-index can be further compressed by using high-order empirical entropy of the text [9, 11].

When compressed suffix arrays and compressed suffix trees were first introduced, they were constructed from suffix arrays and suffix trees, which implies they were not constructed in optimal $O(n \log |\Sigma|)$-bit working space. For constructing compressed suffix arrays in optimal working space, Lam et al. [24] developed an $O(n \log n)$-time algorithm and Hon et al. [16] developed two algorithms one of which runs in $O(n \log \log |\Sigma|)$ time and the other runs in $O(n \log^{\epsilon} n)$ time. For constructing compressed suffix trees, Hon et al. [15, 16] proposed an $O(n \log^{\epsilon} n)$-time algorithm in optimal working space. By this method, one can construct a compressed suffix tree supporting pattern search in $O(m'|\Sigma|)$ time where $m' = m \log^{\epsilon} n$. However, compressed suffix trees supporting pattern search in $O(m' \log |\Sigma|)$ time cannot be constructed by this method.

In this paper, we present a new compressed suffix tree supporting $O(m' \log |\Sigma|)$-time pattern search and its construction algorithm running in $O(n \log^{\epsilon} n)$ time using optimal $O(n \log |\Sigma|)$-bit working space.

– Our compressed suffix tree consists of a compressed suffix array, a succinct representation of lcp information, and a succinct representation of a suffix tree topology. The compressed suffix array is the same as the one developed by Grossi and Vitter [13] and the succinct representation of lcp information is the same as the one developed by Sadakane [29, 30]. Our main contribution is to present a new succinct representation of a suffix tree topology which