String Superprimitivity Test on the Reconfigurable Bus Model [*]

Jenn-Dar Chang
Department of Applied Mathematics

Chang-Biau Yang
Kuo-Si Huang
Department of Computer Science and Engineering,
National Sun Yat-sen University, Kaohsiung, Taiwan
Email:cbyang@cse.nsysu.edu.tw

Abstract

String regularities such as repetition, palindrome, period, seed, square, cover, etc., have been studied extensively recently. Particularly, string regularities play an important role of applications in bioinformatics tools. For a given string, its quasiperiod is the cover with the minimum length, which can be used in the hybridization method in DNA sequencing.

Many algorithms have been proposed to solve some computational problems in $O(1)$ time on the reconfigurable bus model. In this paper, we concentrate to solve the string *superprimitivity* test problem on the reconfigurable bus model. And we propose a parallel algorithm in $O(1)$ time to solve the string superprimitivity test problem on an $n \times n$ CREW reconfigurable mesh.

1 Introduction

The string problems have been studied for many years, but it seems endless because many new problems arise for special uses. String regularities [1,7,12] have many applications in many areas of science, such as coding and automata theory, formal language theory, bioinformatics, etc. In particular, the hybridization DNA sequencing method have to find the string regularities *cover* and *quasiperiod* of the given DNA sequences [9].

A typical string regularity is *square*, which is a substring of a string consisting of exactly two consecutive ones, such as *aa* and *aabaab* in string *aabaabc*. There are many other string regularities, such as repetition, palindrome, period, seed, square, cover, etc. Informally, a string x is *repetition* if it has the form of zz. For examples, *papa* and *mama* are two repetitions. If a string is the same in forward and backward read, such as *wow* and 1234321, then the string is *palindrome.* If all positions of a string x can be covered by some appearances of another string z, we say that string z *covers* string x. For example, *bab* and *babab* cover *bababab.*

Quasiperiod, defined by Apostolico and Ehrenfeucht [2], is a cover with the smallest length of a string. In the above example, *bab* is the quasiperiod and cover of the string *bababab* and *babab* is only a cover. If a string is only covered by itself, the string is *superprimitive*, such as *abcd* and *abaca*. Apostolico et al. [2] presented an algorithm that all maximal quasiperiodic substrings of a string with length n can be detected in $O(n \log^2 n)$ time with linear auxiliary space. They adopted suffix tree, which is an effective data structure for treating strings. On the cover problem, the all-covers problem [14] and the k-covering problem [8] have also been studied.

Apostolico et al. [3] and Breslauer [4] proposed sequential algorithms in linear time to test whether a string is *superprimitive.* Apostolico et al. [3] used the FL table to test superprimitivity. In the FL table, $FL(i)$ is the length of the border of the ith prefix $x_1 x_2 \cdots x_i$ of the given string x. The table FL is a well-known tool for fast string searching strategies (See [11].), and it is sometimes called *failure function.* Breslauer [4] proposed an *on-line* superprimitivity test algorithm which uses Knuth-Morris-Pratt string matching algorithm [11]. The on-line algorithm tests if each prefix of the input string is superprimitive when the prefix is given.

For the parallel case, Breslauer [5] presented an CRCW-PRAM algorithm to test string superprimitivity in $O(\alpha(n) \log \log n)$ time with $\frac{n}{\alpha(n) \log \log n}$ pro-

[*]This research work was partially supported by the National Science Council of the Republic of China under contract NSC-89-2218-E-110-021.

cessors, where $\alpha(n)$ is the inverse of Ackermann function and n is the length of the given string. The algorithm uses the concept of Breslauer and Galil [6] to solve several string matching problems simultaneously, and then combines the results to answer the superprimitivity problem. And Breslauer [5] also proposed a CRCW-PRAM algorithm in $O(\log \log n)$ time with $O(n \log n / \log \log n)$ processors.

In the two dimensional case, the smallest square submatrix Q of the given matrix A that covers A is said to be the *quasiperiod* of A. If there is no submatrix that covers A, then A is *superprimitive*. Iliopoulos et al. [10] presented an algorithm with $O(\log \log n)$ time for superprimitivity testing of a square matrix on the common CRCW-PRAM model.

In this paper, we shall propose a parallel CREW algorithm that tests whether a string is superprimitive in $O(1)$ time on the reconfigurable bus model with $n \times n$ processors, where n is the length of the given string. If the string is not superprimitivity, our algorithm shall report the quasiperiod of the string.

The rest of this paper is organized as follows. In Section 2, we shall introduce some definitions of string regularities and the reconfigurable bus model. In Section 3, we shall propose a parallel algorithm with $O(1)$ time for solving the string superprimitivity test problem. And finally, a conclusion will be given in Section 4.

2 Preliminary

2.1 Basic Definitions and Some Properties

Some basic definitions of string regularities [2–4] will be given in this subsection. There are many string regularities, we will introduce those we used in superprimitivity test.

Suppose that x, w, u, v are strings, then we have some definitions as follows:

Definition 1 *If $x = w^k$ implies $k = 1$, then x is* primitive.

Definition 2 *If $x = w^c w'$ for some constant $c > 0$ and w' is a prefix of w, then w is a* period *of x.*

For example, *ababc* is primitive, *abc* is a period of *abcabcabcab*, and *ab* is a prefix of *abc*.

Definition 3 *A string w* covers *a string x if for every position $i \in \{1, \cdots, |x|\}$ of x there exists an occurrence of w starting at some position j of x such that $1 \leq j \leq i \leq j + |w| - 1 \leq |x|$.*

Definition 4 *A substring w of x is a* seed *of x if there exists a superstring of x, which is constructed by concatenations and superpositions of w.*

It is clear that every string is covered by itself, which is called a trivial cover. For example, *babab* is covered by *bab*, *babbabab* is covered by *bab*, and *babbababa* has no nontrivial cover but *bab* is a seed of *babbababa*.

Definition 5 *A string x is* superprimitive *if it is covered only by itself. If x is covered by a string w, $w \neq x$, then x is called* quasiperiodic. *If x is quasiperiodic and x is covered by w with minimum length, then w is called the* quasiperiod *of x, which is denoted as $Q(x)$.*

By the above definition, a string x is either superprimitive or quasiperiodic. For example, *bab* is superprimitive and also primitive, *babbabbabba* is primitive , but it is not superprimitive, since *babba* covers *babbabbabba*, *babba* is surely the quasiperiod of *babbabbabba*. As another example, both *babab* and *bab* cover *bababab*, and *bab* is the quasiperiod of *bababab*. Clearly, if a string has only the trivial border, such as *abc*, then it is superprimitive.

Definition 6 *A non-empty substring w is a* border *of string x if x starts and ends with an occurrence of w. That is, $x = wv$ and $x = uw$ for some possibly empty strings u and v.*

For example, *abcd* is a border of *abcdaabcd*. Clearly, a string is always a border of itself. This border is called the *trivial border*. We next give some properties which are used in the superprimitivity testing algorithm. Their proofs can be found in some previously published papers [2–4].

Property 1 *If w is a border of x and q is the quasiperiod of x, then q covers w.*

Property 2 *A string x has a period of length p, $p < |x|$, if and only if it has a non-trivial border of length $|x| - p$.*

Property 3 *If a string w covers a string x, then w is a border of x.*

Figure 1 shows an example of Property 3.

Property 4 *If a string w is a border of x such that $2|w| \geq |x|$, then w covers x.*

Property 5 *If u and v cover a string x, and $|u| \leq |v|$, then u covers v. Hence every string has a unique quasiperiod.*

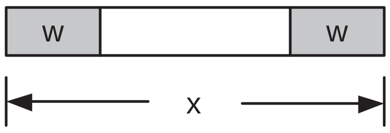

Figure 1: The border w of string x.

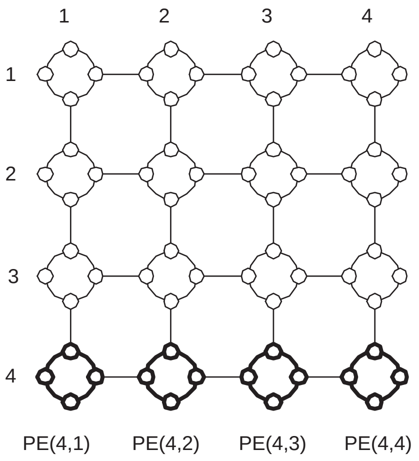

Figure 2: A 4×4 reconfigurable bus model.

Property 6 *If $B(x)$ is a border of string x, and $|Q(x)| \leq |B(x)|$, then $Q(x)$ covers $B(x)$.*

Property 7 *If $B(x)$ is a border of string x, and $|Q(x)| \leq |B(x)|$, for any quasiperiod $Q(B(x))$ of $B(x)$, $Q(x) = Q(B(x))$.*

2.2 The Reconfigurable Bus Model

The model we use in this paper is the *reconfigurable mesh* [13], which consists of two dimensional array of $n \times n$ identical processors with reconfigurable buses. The processor located at (i, j) is referred as $PE(i, j)$, where $1 \leq i \leq n$, $1 \leq j \leq n$. A 4×4 reconfigurable bus model is shown in Figure 2.

Each processor (PE) has its owner locally controllable bus switches for internal connection. These bus switches for internal connection among the four ports of a PE are denoted by N, E, W, and S, and they can be configured during the execution step of algorithms. For example, $\{NEW, S\}$ denotes a PE with its N, E, W ports connected, and $\{NS, WE\}$ denotes a PE with its N-S ports and W-E ports are connected separately. Figure 3 shows the 15 possible connection configurations.

The switches allow the bus to be divided into a number of *subbuses*, the maximal subsets of PE connected in the reconfigurable bus system. A signal sent by any port is received by all other ports connected to it in a PE simultaneously. For example, if each processor connects its N port and S port as show in Figure 4, then all of the processors in a column are able to read the data which is broadcast onto the bus by a particular processor at the same time, and each column is a subbus. The model allows current reading from a bus, requires exclusive writing to a bus (CREW), and assumes that a constant time communication delay on arbitrarily connected bus components.

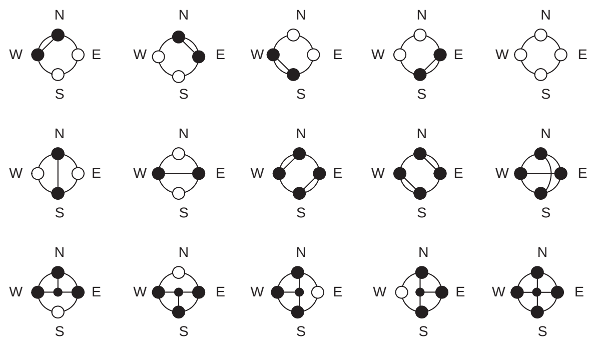

Figure 3: 15 possible connection configurations.

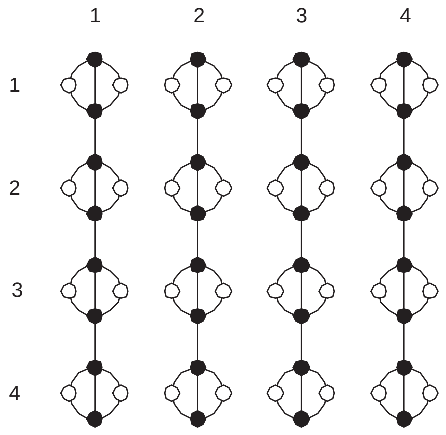

Figure 4: Switches set for column broadcasting.

Each processor in the system is capable of performing the basic arithmetic and logic operations, and has a small fixed local memory. In this paper, we assume that each broadcast requires $\Theta(1)$ time, and each PE performs an arithmetic, logic, control, and switching or communicating operation in constant time. We also assume that each processor knows its position in this model. It allows multiple processors to broadcast data on the different subbuses simultaneously at a time unit.

3 A Parallel Algorithm

In this section, we present a parallel algorithm with $O(1)$ time for solving the string superprimitivity test problem on an $n \times n$ reconfigurable bus model.

Fact 1 *Let $B_1(x)$ and $B_2(x)$ be two borders of a string x, if $|B_1(x)| \leq |B_2(x)|$, then $B_1(x)$ is a border of $B_2(x)$.*

Fact 2 *The quasiperiod $Q(x)$ is one of the borders of string x.*

Based on the above two facts, our algorithm can be divided into two parts :

1. Find all non-trivial borders of the given string.

2. Test whether one of the non-trivial borders covers the given string.

Initially, we assume that the input string $x = x_1 x_2 \cdots x_n$, and each processor $PE(1, j)$, $1 \leq j \leq n$, stores an element x_j.

Algorithm : String Superprimitivity Test

Input: A given string $x = x_1 x_2 \cdots x_n$.

Output: x is superprimitive or x is quasiperiodic.

Procedure FIND: Finding all non-trivial borders of the given string.

Step 1: Establish the column subbuses system. It can be done by every processor connecting its N and S ports. (See Figure 4.)

Step 2: $PE(1, j)$, $1 \leq j \leq n$, which is on the first row, broadcasts the holding element x_j through the column subbuses system. After Step 2, $PE(i, j)$, $1 \leq i \leq n, 1 \leq j \leq n$ receives element x_j from its N port.

Step 3: Establish the diagonal subbuses system as follows. (See Figure 5.)

$PE(1, j)$, $1 \leq j \leq n-1$, and $PE(i, n)$, $1 \leq i \leq n$, connects its W and S ports; $PE(k, k-1)$, $2 \leq k \leq n$, connects its N and E ports; processor $PE(i, j)$, $2 \leq i \leq j \leq n-1$, connects its W and S ports, N and E ports. Other processors do nothing.

Step 4: $PE(1, j)$, $1 \leq j \leq n$, broadcasts element x_j through the diagonal subbuses system. $PE(i, j)$, $1 \leq i \leq j \leq n$ stores element x_{j-i+1} received from its W port. After Step 4, $PE(i, j), 1 \leq i \leq j \leq n$, stores x_j and x_{j-i+1}.

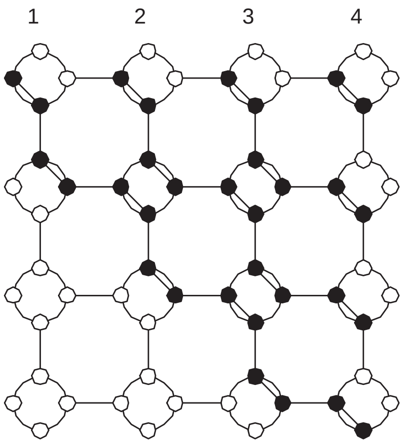

Figure 5: The diagonal subbuses system in Step 3 of Procedure FIND.

Step 5: In $PE(i, j)$, $1 \leq i \leq j \leq n$, if $x_j = x_{j-i+1}$ then W and E ports are connected; otherwise N and S ports are connected.

Step 6: $PE(i, i)$, $2 \leq i \leq n$, tries to send a border finding signal to $PE(i, n)$. If $PE(i, n), 2 \leq i \leq n$ can receive this signal from its W port, then it sets border signal b_i to 1, otherwise sets b_i to 0. If b_i is 1, $x_i x_{i+1} ... x_n$ is a border of string x.

Step 7: In $PE(i, n)$, $2 \leq i \leq n$, and if b_i is 1, connect its N port and E port; otherwise its N port and S port are connected.

Step 8: Send a maximal border finding signal from S port of $PE(1, n)$. If the signal is received by E port of a $PE(i, k)$, then the string $x_k x_{k+1} \cdots x_n$ is the border with the maximal length, and its length is $m = n - k + 1$. If there does not exist such $PE(i, k)$ and m, the string x has no non-trivial border, and it is superprimitivity.

Step 9: Connect N port and S port for all $P(i, n)$, then broadcast m to all $P(i, n)$.

After we find all non-trivial borders and the longest non-trivial border with length m. If there exists at least one non-trivial border, we will test whether one of these borders covers x and find the quasiperiod.

Procedure TEST: Testing whether one of the non-trivial borders covers x.

Step 1: Establish the diagonal subbus system mentioned at Step 3 in Procedure FIND. (See Figure 5.) $PE(i, n)$, $2 \leq i \leq n$, broadcasts b_i and m through the diagonal subbuses system. After Step 1, $PE(i, j)$, $1 \leq i \leq j \leq n$, receives b_{n-j+i} and m.

Step 2: In $PE(i,j)$, $1 \leq i,j \leq n$, connect its N and E ports.

Step 3: In $PE(i,j)$, $1 \leq i \leq n$ and $i \leq j \leq n-1$, if $x_j = x_{j-i+1}$ and $b_{n-j+1} = 0$, then $\{NEW\}$ are connected; if $x_j = x_{j-i+1}$ and $b_{n-j+1} = 1$, then $\{NEWS\}$ are connected; otherwise N and S ports are connected. In $PE(i,j)$, $1 \leq i \leq n$ and $j = n$, if $x_j = x_{j-i+1}$, then connect its W and E ports, else connect its N and S ports.

Step 4: $PE(1,1)$ sends a signal through the bus system.

Step 5: In $PE(i,n), 1 \leq i \leq n$, if the processor receives this signal from its W port, then E and S ports are connected; otherwise N and S are connected.

Step 6: Try to send a quasiperiod finding signal from port S of $PE(n,n)$ to $PE(1,n)$. If there exists $PE(i,n), 1 \leq i \leq n$, whose E port receives the signal, then the length of quasiperiod is $n-i+1$, and $x_i x_{i+1} \cdots x_n$ is the quasiperiod of string x.

After we test whether one of these borders covers the given string x, if this procedure does not output the result that x is quasiperiodic, then algorithm outputs the result that x is superprimitive. The quasiperiod is also reported by the algorithm.

Theorem 1 *Let $B(x)$ be a border of a string x, if the quasiperiod $Q(x)$ is not covered by $B(x)$, i.e. $|Q(x)| \geq 2|B(x)|$, then the test signal for $PE(1,|B(x)|)$ will not be received by $PE(|Q(x)|-|B(x)|, |Q(x)|-|B(x)|)$.*

Proof: $B(x)$ and $Q(x)$ are borders of x, and $|Q(x)| \geq 2|B(x)|$, by Fact 1, $B(x)$ is a border of $Q(x)$. It implies that the head and tail of $Q(x)$ with length $|B(x)|$ is covered by $B(x)$. There must exist at least one substring whose length is larger than 1 such that $B(x)$ does not cover the substring. In other words, the substring is a break point such that $B(x)$ cannot cover $Q(x)$. The break point must locate in the middle part of $Q(x)$, because the head and tail are covered by $B(x)$ and the signal cannot pass through the break point. If $PE(|Q(x)|-|B(x)|, |Q(x)|-|B(x)|)$ can receive the signal of $PE(1,|B(x)|)$, then by the definition of cover, $Q(x)$ will be covered by $B(x)$ since $Q(x)-B(x)$ is covered by $B(x)$ and the tail of $Q(x)$ is $B(x)$ which is covered by $B(x)$ itself. Then $Q(x)$ is not the quasiperiod, and it contradict the assumption. □

Theorem 2 *Let $B(x)$ be a border of a string x, if the quasiperiod $Q(x)$ covers $B(x)$, then the test signal from $PE(1,|Q(x)|)$ will be received by $PE(|B(x)|,|B(x)|)$.*

Proof: Since $Q(x)$ covers $B(x)$, the tail of $B(x)$ is $Q(x)$. Then signal from $PE(1,|Q(x)|)$ will be received by $PE(|B(x)|,|B(x)|)$ via the path formed by covers. □

Theorem 3 *Let $B(x)$ be a border of a string x, if the quasiperiod $Q(x)$ is not covered by $B(x)$, i.e. $|Q(x)| \geq 2|B(x)|$, then the test signal for $PE(i, j+|B(x)|)$ will not be received by $PE(i+|Q(x)|-|B(x)|, j+|Q(x)|-|B(x)|)$, where $i+|Q(x)|-|B(x)| \leq n$ and $j+|Q(x)|-|B(x)| \leq n$.*

Proof: If the signal can be sent to $PE(i, j+|B(x)|)$, then the string $x_1 x_2 \cdots x_j$ is covered by $Q(x)$. We can disregard the prefix part $x_1 x_2 \cdots x_j$ because it is covered by $Q(x)$. Then it will become the general case of Theorem 1. □

Theorem 4 *Let $B(x)$ be a border of a string x, if the quasiperiod $Q(x)$ covers $B(x)$, then the test signal from $PE(i, j+|Q(x)|)$ will be received by $PE(i+|B(x)|, j+|B(x)|)$, where $i+|B(x)| \leq n$ and $j+|B(x)| \leq n$.*

Proof: If the signal can be sent to $PE(i, j+|Q(x)|)$, then the string $x_1 x_2 \cdots x_j$ is covered by $Q(x)$. We can disregard the prefix part $x_1 x_2 \cdots x_j$ because it is covered by $Q(x)$. Then it will become the general case of Theorem 2. □

Corollary 1 *If $PE(i,n)$, $1 \leq i \leq n$ receives one test signal, then it must be the signal sent by the quasiperiod $Q(x)$.*

Proof: By Theorems 1, 2, 3 and 4, the signal of $B(x)$ that does not cover $Q(x)$ would be terminated by some break point before it arrives at $Q(x)$. If the signal is sent from some border $B(x)$, then the PE receives the signal must be the signal from the minimal cover, which is the quasiperiod. So, we can disregard the signal sent by each of the borders whose length is longer than the quasiperiod. □

Theorem 5 *There is an algorithm to test superprimitivity and find the quasiperiod in O(1) time on a reconfigurable mesh.*

Proof: Since the switch configuration, signal broadcasting and sending of each step in our algorithm takes constant time, and there is no loop in our algorithm, the algorithm can be done in $O(1)$ time. □

Now, we shall give two examples for illustrating our algorithm. Note that we omit some switches of processors in some figures if there is no confusion.

Example 1: $x = ababa$, where $|x| = 5$. Figure 6 and Figure 7 show how the algorithm works with this example.

In Figure 6, we get two non-trivial borders a, aba and $m = 3$. Then, the algorithm will check whether a or aba covers x, as shown in Figure 7.

Since processor $PE(3,3)$ receives the signal from its W port which is sent by $PE(1,1)$, the border aba, which has the maximal length, covers the given string $abaababababa$ and then the algorithm outputs the result that x is quasiperiodic. Surely, the input string is not superprimitive.

Example 2: $x = abdab$, where $|x| = 5$. Some execution steps are shown in Figure 8 and Figure 9.

In Figure 8, we get only one non-trivial border ab with length $m = 2$. Then, the algorithm will check whether it covers x, as shown in Figure 9.

Since each processor $PE(i,5), 1 \leq i \leq 5$ does not receive the signal from its W port which is sent by $PE(1,1)$, the border ab does not covers the given string $abdab$ and algorithm outputs the result that x is superprimitive.

4 Conclusion

In this paper, we give a parallel CREW algorithm with $O(1)$ time to solve the string superprimitivity test problem on an $n \times n$ reconfigurable bus model.

In the future, the string superprimitivity test problem can be extended to find all covers in strings or extended to two dimensional or higher dimensional string superprimitivity test problem with $O(1)$ time on the two dimensional or higher dimensional reconfigurable bus model. Other string regularities may be solved on the reconfigurable bus model. We can also apply these string regularities to other areas, such as the problems of DNA sequences in molecular biology.

References

[1] A. Apostolico, "Optimal parallel detection of squares in strings," *Algorithmica*, Vol. 8, pp. 285–319, 1992.

[2] A. Apostolico and A. Ehrenfeucht, "Efficient detection of quasiperiodicities in strings," *Theoretical Computer Science*, Vol. 119, pp. 247–265, 1993.

[3] A. Apostolico, M. Farach, and C. S. Iliopoulos, "Optimal superprimitivity testing for strings," *Information Processing Letters*, Vol. 39, No. 1, pp. 17–20, 1991.

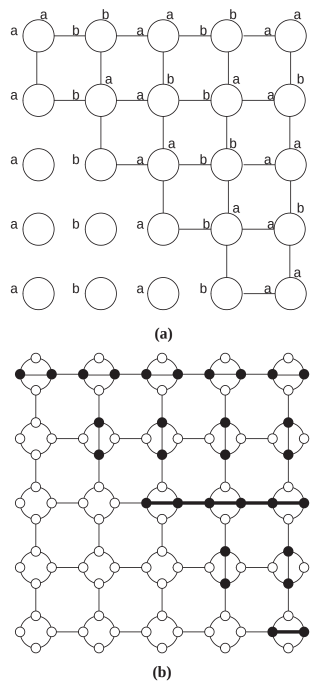

Figure 6: Finding all non-trivial borders for $x = ababa$. (a)After Step 4 in FIND. (b)After Step 6 in FIND.

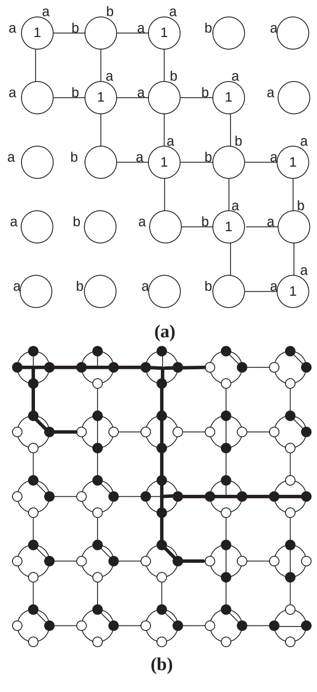

Figure 7: Testing whether one of the non-trivial borders covers $x = ababa$. (a)After Step 1 in TEST. (b)After Step 4 in TEST.

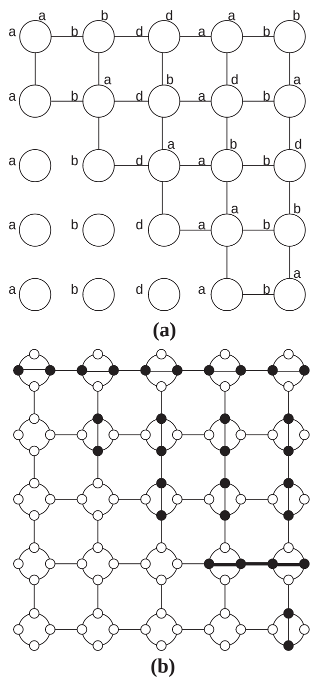

Figure 8: Finding all the non-trivial borders for $x = abdab$. (a)After Step 4 in FIND. (b)After Step 6 in FIND.

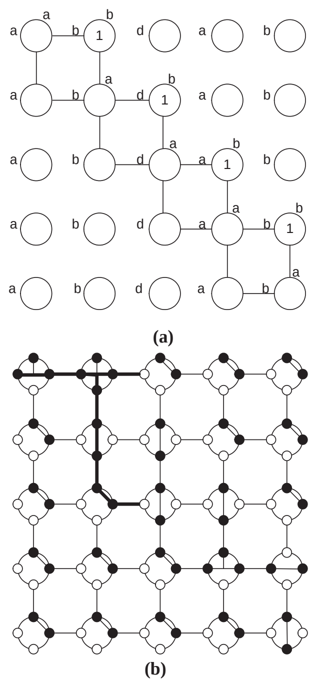

Figure 9: Testing whether the non-trivial border covers $x = abdab$. (a)After Step 1 in TEST. (b)After Step 4 in TEST.

[4] D. Breslauer, "An on-line string superprimitivity test," *Information Processing Letters*, Vol. 44, No. 6, pp. 345–347, 1992.

[5] D. Breslauer, "Testing string superprimitivity in parallel," *Information Processing Letters*, Vol. 49, pp. 235–241, 1994.

[6] D. Breslauer and Z. Galil, "An optimal $o(\log \log n)$ time parallel string matching algorithm," *SIAM Journal on Computing*, Vol. 19, No. 6, pp. 1051–1058, 1990.

[7] D. Breslauer and Z. Galil, "Finding all periods and initial palindromes of a string in parallel," *Algorithmica*, Vol. 14, pp. 355–366, 1995.

[8] M. Crochemore, C. S. Iliopoulos, and M. Korda, "Two-dimensional prefix string matching and covering on square matrices," *Algorithmica*, Vol. 20, pp. 353–373, 1998.

[9] A. Duval and W. Smyth, "Covering a circular string with substrings of fixed length," *Internationl Journal of Foundations of Computer Science*, Vol. 7, No. 1, pp. 87–93, 1996.

[10] C. S. Iliopoulos and M. Korda, "Optimal parallel superprimitivity testing for square arrays," *Parallel Processing Letters*, Vol. 6, No. 3, pp. 299–308, 1996.

[11] D. E. Kunth, J. H. M. Jr, and V. R. Pratt, "Fast pattern matching in strings," *SIAM Journal on Computing*, Vol. 6, No. 2, pp. 323–350, 1977.

[12] M. G. Main and R. J. Lorentz, "An $o(n \log n)$ algorithm for finding all repetitions in a string," *Journal of Algorithms*, Vol. 5, pp. 422–432, 1984.

[13] R. Miller, V. P. Kumar, D. Reisis, and Q. Stout, "Meshes with reconfigurable buses," *Proc. MIT Conf. Advanced Research in VLSI*, pp. 163–178, Apr. 1988.

[14] D. Moore and W. Smyth, "Computing the covers of a string in linear time," *Proc. 5th ACM-SIAM Symposium on Discrete Algorithms*, pp. 511–515, 1994.