Enhancing ERS-A Algorithm for Pattern Matching (EERS-A)

Dima Suleiman¹, Mariam Itriq¹, Aseel Al-Anani², Rola Al-Khalid², Amjad Hudaib²

¹Department of Business Information Technology, King Abdullah II School for Information Technology, The University of Jordan, Amman, Jordan
²Department of Computer Information Systems, King Abdullah II School for Information Technology, The University of Jordan, Amman, Jordan
Email: dima.suleiman@ju.edu.jo, m.itriq@ju.edu.jo, a.anani@ju.edu.jo, r.khalid@ju.edu.jo, ahudaib@ju.edu.jo

Received 25 February 2015; accepted 18 March 2015; published 20 March 2015

Copyright © 2015 by authors and Scientific Research Publishing Inc.
This work is licensed under the Creative Commons Attribution International License (CC BY).
http://creativecommons.org/licenses/by/4.0/

Abstract

Pattern matching is a very important topic in computer science. It has been used in various applications such as information retrieval, virus scanning, DNA sequence analysis, data mining, machine learning, network security and pattern recognition. This paper has presented a new pattern matching algorithm—Enhanced ERS-A, which is an improvement over ERS-S algorithm. In ERS-A, two sliding windows are used to scan the text from the left and the right simultaneously. The proposed algorithm also scans the text from the left and the right simultaneously as well as making comparisons with the pattern from both sides simultaneously. The comparisons done between the text and the pattern are done from both sides in parallel. The shift technique used in the Enhanced ERS-A is the four consecutive characters in the text immediately following the pattern window. The experimental results show that the Enhanced ERS-A has enhanced the process of pattern matching by reducing the number of comparisons performed.

Keywords
Pattern Matching, Enhanced Two Sliding Windows Algorithm, RS-A Fast Pattern Matching Algorithm, Enhanced RS-A

1. Introduction

Many applications use pattern matching algorithms such as search engines, anti-virus and biological applications such as DNA [1]-[5].

Most of the algorithms have been implemented in order to make the searching process faster and more efficient; this can be achieved either by reducing the number of attempts, comparisons or by both. Some of algo-
algorithms made enhancements on preprocessing phase [6]-[9] and searching phase [10]-[12], and others modified the shifting technique [5] [9] [13]-[15] used in a case of a mismatch between the pattern and the text.

Searching process differs from one algorithm to another: some algorithms scan the text from one side only, either from left [15] or from right [5]; other algorithms use two sliding windows to scan the text from the left and the right sides simultaneously [10]-[13].

Some algorithms made modification on the shifting values. The shift value is the amount of shift that the sliding window will move in a case of a mismatch between the text and the window; shift values depend on the number of consecutive characters immediately after the sliding window. Some algorithms use one consecutive character [16] [17]; others use two [10] [12] [15] and few use three [13].

In this paper, we propose a new pattern matching algorithm—Enhanced ERS-A. The algorithm uses two sliding windows such as TSW [12], ETSW [10], ERS-A [11] and EBR [13] algorithms. It also uses four consecutive characters to compute the shift values such as RS-A [5] and ERS-A [11] algorithms. In addition to that, it uses the same comparison technique between the pattern and the text that was used in ETSW [10].

This paper compares between the new algorithms of EERS-A, ETSW [10] and ERS-A [11]. The results showed that the new algorithm is better than others as explained in Section 5. The reminder of this paper is organized as follows: Section 2 consists of related works; Section 3 explains EERS-A algorithm; Section 4 covers the analysis. Finally, the conclusion and the future work are presented in Section 6.

2. Related Works

Pattern matching algorithms were needed in many applications [2] [14] [18]-[20]. Some algorithms made enhancement in a memory used in preprocessing phase, while others try to make the searching process faster and more efficient [8] [9] [21].

The Berry-Ravindran algorithm (BR) [15] made enhancement in Boyer Moore algorithm [16]. In a case of a mismatch between the text and the pattern window Boyer uses one consecutive character in the text immediately to the right of the pattern window to determine amount of shift the window must move, on the other hand Berry-Ravindran uses two consecutive characters.

Many enhancements made in Berry-Ravindran algorithm (BR) [15], some algorithms changed the searching process but on the other hand they used the same bad character shift values, bad character shift depends on using two consecutive characters immediately that follow the text such as TSW [12] and ETSW [10]. Others made enhancement on the shift values such as RS-A [5] and ERS-A [11] these two algorithms changed the shift values by using four consecutive character instead of two, also EBR [13] algorithm made modifications on Berry-Ravindran bad character shift by depending on using three consecutive characters.

Two Sliding Windows algorithm TSW [12] used the same preprocessing technique used in BR [15] but made enhancements on the searching phase. In a case of a mismatch between the pattern and text, it uses two consecutive characters to determine the amount of shift the pattern window must slide which is the same technique used in BR [15], but instead of using one pattern window to scan the text as in BR [15] it uses two sliding windows that scan it in parallel.

Searching process become faster in Enhanced Two Sliding Window algorithm (ETSW) [10]. ETSW [10] made enhancements on TSW [12] by minimizing the number of comparisons needed but it doesn’t make any changes on the number of attempts. The reason for this is that two algorithms used the same preprocessing techniques and also used the two sliding window, the only difference between them is related to the idea that TSW [12] compares the text with the pattern from one side of the pattern while ETSW [10] compare the text with the pattern from the both sides of the pattern at the same time, the best time complexity is \( O(m/2) \) and the worst case time complexity is \( O\left(\left(\left(\frac{n}{2} - \frac{m}{2} + 1\right)\frac{m}{2}\right)\right) \). The preprocess time complexity is \( O(2(m-1)) \).

RS-A [5] algorithm used only one window to search for a pattern \( p \) in a text \( t \) from the right side of the text. In order to make the searching process faster ERS-A [11] algorithms made enhancements on RS-A by using two windows instead of one. The left window aligned with the text from the left and in a case of a mismatch the window will be shifted to the right, and the right window aligned with text from the right and in a case of a mismatch the window will be shifted to the left. The two windows slide in parallel. The search will stop either when the pattern is found or in a case the pattern not found at all.

The EERS-A made enhancement on ERS-A [11], it uses the same preprocessing technique that depends on using four consecutive characters to determine the amount of shift, it also uses two sliding windows to scan the text from the left and right sides at the same time. Enhancements made on a comparison between the text and the
pattern, in this case it uses the same method used in ETSW [10], which make comparisons between the text and the pattern from both sides of the pattern at the same time. It is obvious from the results that there are no changes in average number of attempts in both ERS-A [11] and EERS-A, on the other hand there are a clear differences in average number of comparisons. The EERS-A is more efficient and faster.

3. The Enhanced ERS-A Algorithm

EERS-A algorithm improved the searching process by scanning the pattern as well as the text from both sides simultaneously. EERS-A used two sliding windows to scan the text from both sides at the same time; also comparisons between the pattern and the text happened from both sides of the pattern.

EERS-A algorithm used the same searching technique used in ERS-A, they uses two sliding windows to search for a pattern p in a text t. Two sliding windows scan the text from the left and right side at the same time and in a case of a mismatch both algorithms will use RS-A bad character shift function [5]. The searching process will stop either when a pattern is found or in a case a pattern is not found in the text at all.

The main difference between EERS-A and ERS-A [11] algorithms is that in ERS-A [11] comparisons between the pattern and the text happened only from the left side of the pattern while in EERS-A comparisons done from left and right sides of the pattern at the same time.

4. Pre-Processing Phase

Two arrays are generated in this phase nextl and nextr, each array is one-dimensional array. The values of the nextl array are calculated according ERS-A algorithm [11]. Initially the indexes of the four consecutive characters of the text after aligning it with the right window are (n – m – 4), (n – m – 3), (n – m – 2) and (n – m – 1) for a, b, c and d respectively as in Equation (2).

\[
\text{Left shift value } [a,b,c,d] = \begin{cases} 
1 & \text{if } p[m-1] = a \\
2 & \text{if } p[m-2][m-1] = ab \\
3 & \text{if } p[m-3][m-2][m-1] = abc \\
m+1 & \text{if } p[0] = b \\
m+2 & \text{if } p[0] = c \\
m+3 & \text{if } p[0] = d \\
m-i & \text{if } p[i][i+1][i+2][i+3] = abcd \\
m+4 & \text{otherwise}
\end{cases}
\]

On the other hand, the values of the nextr array are calculated according RS-A algorithm [5]. The shift values are calculated by using four consecutive text characters a, b, c and d which are used to calculate the shift values in a case of a mismatch from the right side of the text as in Equation (1).

\[
\text{Right shift value } [a,b,c,d] = \begin{cases} 
m+3 & \text{if } p[m-1] = a \\
m+2 & \text{if } p[m-1] = b \\
m+1 & \text{if } p[m-1] = c \\
1 & \text{if } p[0] = d \\
2 & \text{if } p[0][1] = cd \\
3 & \text{if } p[0][1][2] = bcd \\
m-(m-4-i) & \text{if } p[i][i+1][i+2][i+3] = abcd \\
m+4 & \text{otherwise}
\end{cases}
\]
The pre-processing phase is the same in both ERS-A and EERS-A algorithms but the searching phase is enhanced in EERS-A.

5. Searching Phase

In EERS-A algorithm the searching process started by aligning two windows with the text, the left window aligned with the beginning of the text and the right window aligned with the end of the text. In case of a mismatch the proposed window will be shifted according to next array values calculated in a preprocessing phase; the left window will be shifted to the right and the right window will be shifted to left.

Four pointers will be used in a comparison process between the text and the pattern, two for each window. The left window uses the L and temp_newlindex pointers while the right window uses the R and temp_newrindex pointers.

Comparisons between the text and the pattern from both sides is the same, in each window the first character of the pattern is compared with the corresponding text character and at the same time the last character of the pattern is compared with the corresponding character in the text. Each window uses different pointers as explained below:

6. Left_Window Search Process

After aligning the left window with the text from the left, comparisons between the pattern and the text will be done using two pointers, L and temp_newlindex, L pointer points at the last character of the pattern and the temp_newlindex points at the first character of the pattern, comparisons are made between the pointers and the corresponding characters of the text.

If a mismatch occurs in any pointer a shift will occur according to the ERS-A bad character algorithm. In case of a match the two pointers will move. The L pointer will move to the left and the temp_newlindex will move to the right. A movement of pointers in a case of a match will stop either when two pointers reach the middle of the pattern or when the L pointer is less than or equal the temp_newlindex, in either case the pattern is found.

7. Right_Window Search Process

After aligning the right window with the text from the right, comparisons between the pattern and the text will be done using two pointers, R and temp_newrindex, R pointer points at the first character of the pattern and the temp_newrindex points at the last character of the pattern, comparisons are made between the pointers and the corresponding characters of the text.

If a mismatch occurs in any pointer a shift will occur according to the ERS-A bad character algorithm. In case of a match the two pointers will move. The R pointer will move to the right and the temp_newrindex will move to the left. A movement of pointers in a case of a match will stop either when two pointers reach the middle of the pattern or when the R pointer is less than or equal the temp_newrindex, in either case the pattern is found.

The proposed EERS-A algorithm is explained in Figure 1.

8. Working Example

In this section, we will give an example to explain the new algorithm.

Given:

Pattern (P) = “ACCBCBAC”, \( m = 8 \),
Text (T) = “ABCDABADCABACBAADBCEDABADDDDBACCBACBCBAB”, \( n = 50 \).

9. Pre-Processing Phase

Initially, \( shiftl = shiftr = m + 4 = 12 \).

The shift values are stored in two arrays nextl and nextr as shown in Figure 2(a) and Figure 2(b) respectively.

To build the two next arrays (nextl and nextr), we take each four consecutive characters of the pattern and
To search in a text, we use two windows (left windows, right window)

Align the left window with the text from the left

Align the right window with the text from the right

Stop the search, pattern not found

While the two windows not overlapped

Yes

No

Two pointers will be used L and temp_newrindex, L
pointer point at the last character of the pattern and the
temp_newrindex point at the first character of the pattern.

Two pointers will be used L and temp_newrindex, R
pointer point at the first character of the pattern and the
temp_newrindex point at the last character of the pattern.

Comparisons are made between the pointers and the
 corresponding characters of the text

Comparisons are made between the pointers and the
corresponding characters of the text

No

No

Match

Yes

Yes

In case a mismatch occurs in any pointer,
the left window will shift to the right
according to the ERS-A bad character
algorithm Equation (2)

In case a mismatch occurs in any pointer,
the right window will shift to the left
according to the ERS-A bad character
algorithm Equation (1)

Two pointers will move. The L pointer
will move to the left and the
temp_newrindex will move to the right

Two pointers will move. The R pointer
will move to the right and the
temp_newrindex will move to the left

If two pointers reach
the middle or L
pointer is less than or
equal the

If two pointers reach
the middle or R
pointer is less than or
equal the

Yes

Yes

Pattern match with the text from the left

Pattern match with the text from the right

Figure 1. EERS-A flowchart.

<table>
<thead>
<tr>
<th>Index</th>
<th>Shift Values from the left</th>
</tr>
</thead>
<tbody>
<tr>
<td>nextl</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>8</td>
</tr>
</tbody>
</table>

(a)

<table>
<thead>
<tr>
<th>Index</th>
<th>Shift Values from the right</th>
</tr>
</thead>
<tbody>
<tr>
<td>nextr</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>4</td>
</tr>
</tbody>
</table>

(b)

Figure 2. The nextl and nextr arrays.
give it an index starting from 0. For example for the pattern structure ACCBCBAC, the consecutive characters
ACCB, CCBC, CBCB, BCBA and CBAC are given the indexes 0, 1, 2, 3 and 4 respectively.

The shift values for the nextr array are calculated according to Equation (1) while the shift values for the nextl
array are calculated according to Equation (2).

10. Searching Phase

The searching process for the pattern P is explained through the working example as shown in Figure 3.

10.1. First Attempt

In the first attempt (see Figure 3(a)), we align the left sliding window with the text from the left. In this case, a
comparison is made between the text character located at index 0 (character A) with the leftmost character in the
pattern (character A) although a match occurs, the total result is a mismatch since at the same time a comparison
must be made between the text character at index 7 (character D) with the rightmost character in the pattern
(character C); therefore we take the four consecutive characters of the text at index 8, 9, 10 and 11 which are (B, A, C and E) respectively. To determine the amount of shift \( \text{shiftl} \) we have to do the following:

Since if \( p[m \rightarrow 3]-m \rightarrow 2 \rightarrow m \rightarrow 1 = abc \); BAC = BAC then according to a preprocessing algorithm the shift
value will be 3.

10.2. Second Attempt

In the second attempt (see Figure 3(b)), we align the right sliding window with the text from the right. In this
case, a match occurs between text character at index 42 (A) and leftmost character of the pattern character A but
a mismatch occurs between the last text character B with the last pattern character C so as a total result,
there is a mismatch; therefore we take the four consecutive characters from the text at index 38, 39, 40 and 41 which are A, C, C and B respectively. To determine the amount of shift ($shift_r$), we have to do the following:

a) We find the index of ACCB in the pattern which is 0.

b) Since we search the text from the right side we use $next_r$ array, and $shift_r = next_r[0] = 4$; therefore the window will be shifted to the left 4 steps.

10.3. Third Attempt

In the third attempt (see Figure 3(c)), a mismatch occurs from the left between the text character at index 3 (character D) and the leftmost character in the pattern (character A) while there is a match between the text character at index 10 (character C) and the rightmost character in the pattern (character C); therefore we take the four consecutive characters from the text at indexes 11, 12, 13 and 14 which are E, D, A and B respectively, since EDAB is not found in the pattern, so the window will be shifted to the right 12 steps.

10.4. Fourth Attempt

In the fourth attempt (see Figure 3(d)), a comparison is made between the text character located at index 38 (character A) with the leftmost character in the pattern (character A) at the same time a comparison must be made between the text character at index 45 (character B) with the rightmost character in the pattern (character C) since there is a mismatch we take the four consecutive characters of the text at index 34, 35, 36 and 37 which are D, D, D and A respectively; since DDDB is not found in the pattern, so the window will be shifted to the left 12 steps.

10.5. Fifth Attempt

We align the left most character of the pattern $P[0]$ with $T[15]$. A comparison between the pattern and the text characters leads to a complete match at index 15. In this case, the occurrence of the pattern is found using the left window.

11. Analysis

**Preposition 1**: The space complexity is $O(2(m - 3))$ where $m$ is the pattern length.

**Preposition 2**: The pre-process time complexity is $O(2(m - 3))$.

**Lemma 1**: The worst case time complexity is $O(\left((n/2 - m/2 + 1)\right)(m/2))$.

**Proof**: The worst case occurs when a match between the text and patterns occurs in all characters except a character at index $m/2$ in the pattern, and if at the same time the shift values in case of a mismatch equal to 1.

**Lemma 2**: The best case time complexity is $O(m/2)$.

**Proof**: The best case occurs when the pattern is found in the leftmost or the rightmost sides of the text.

**Lemma 3**: The average case time complexity is $O(n/(2*(m/2 + 4)))$.

**Proof**: The average case occurs when the four consecutive characters of the text directly following the sliding window is not found in the pattern. In this case, the shift value will be $(m/2 + 4)$ and hence the time complexity is $O\left(n/(2*(m/2 + 4))\right)$.

12. Experimental Results and Discussion

Several experiments have been done using Book 1 from the Calgary corpus [22] to be the text in most of searching algorithms. Book 1 consists of 141,274 words (752,149 characters). Book 1 consists of 141,274 words (752,149 characters). Patterns of different lengths are also taken from Book 1.

**Table 1** shows the results of comparing the algorithms ETSW, ERS-A and EERS-A.

In **Table 1**, the first column related to the pattern length; second column is the number of words in a certain length. It can be clearly seen that the number of attempts in EERS-A and ERS-A are the same since the two algorithms use the same shifting techniques in a case of a mismatch, and according to the results it’s clear that the number of comparisons in EERS-A are better than all other algorithms. For example, as shown in **Table 1**, 2896 words of length 6, the average number of comparisons in ETSW is 7633, in ERS-A is 6750 and in the new algorithms is 6212, which is the minimum value among the others values.
ETSW algorithm and EERS-A algorithm use the same comparison technique, they use two sliding windows to scan the text, and in addition to that they use the same technique when comparing the text with the pattern where comparisons happened from both sides of the pattern simultaneously. It is clear that the number of comparisons and attempts in EERS-A algorithm are better than ERS-A algorithm and this refers to using different shifting technique. While ETSW uses Berry-Ravindran (BR) [15] bad character shift values which uses two consecutive characters, EERS-A uses ERS-A [11] bad character shift values that uses four consecutive characters.

Table 2 shows the average number of attempts and comparisons for 100 words taken from the right side of Book1. It is clear that EERS-A is the best among all others according to the number of comparisons and the same results can be seen in Table 3 and Table 4.

Table 5 and Figure 4 show the average number of comparisons needed to search for patterns with different lengths. The results show that EERS-A finds the patterns with minimum number of comparisons compared with

### Table 1. The average number of attempts and comparisons of ETSW, ERS-A and EERS-A algorithms.

<table>
<thead>
<tr>
<th>Pattern length</th>
<th>Number of words</th>
<th>ETSW</th>
<th>ERS-A</th>
<th>EERS-A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Attempts</td>
<td>Comparisons</td>
<td>Attempts</td>
</tr>
<tr>
<td>5</td>
<td>4535</td>
<td>4456</td>
<td>3549</td>
<td>3533</td>
</tr>
<tr>
<td>6</td>
<td>2896</td>
<td>7596</td>
<td>7633</td>
<td>6166</td>
</tr>
<tr>
<td>7</td>
<td>1988</td>
<td>9341</td>
<td>9118</td>
<td>7737</td>
</tr>
<tr>
<td>8</td>
<td>1167</td>
<td>10056</td>
<td>10115</td>
<td>8451</td>
</tr>
<tr>
<td>9</td>
<td>681</td>
<td>9538</td>
<td>9590</td>
<td>8106</td>
</tr>
<tr>
<td>10</td>
<td>382</td>
<td>9283</td>
<td>9339</td>
<td>7970</td>
</tr>
<tr>
<td>11</td>
<td>191</td>
<td>5451</td>
<td>5482</td>
<td>4701</td>
</tr>
<tr>
<td>12</td>
<td>69</td>
<td>6384</td>
<td>6433</td>
<td>5589</td>
</tr>
<tr>
<td>13</td>
<td>55</td>
<td>7947</td>
<td>7986</td>
<td>6955</td>
</tr>
<tr>
<td>14</td>
<td>139</td>
<td>19437</td>
<td>19535</td>
<td>17115</td>
</tr>
<tr>
<td>15</td>
<td>32</td>
<td>19682</td>
<td>19782</td>
<td>17385</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>20029</td>
<td>20092</td>
<td>17722</td>
</tr>
<tr>
<td>17</td>
<td>3</td>
<td>21897</td>
<td>22147</td>
<td>19521</td>
</tr>
</tbody>
</table>

### Table 2. The average number of attempts and comparisons performed to search for (100) patterns selected from the right side of the text.

<table>
<thead>
<tr>
<th>Pattern length</th>
<th>Number of words</th>
<th>ETSW</th>
<th>ERS-A</th>
<th>EERS-A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Attempts</td>
<td>Comparisons</td>
<td>Attempts</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>185</td>
<td>187</td>
<td>146</td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>227</td>
<td>230</td>
<td>182</td>
</tr>
<tr>
<td>7</td>
<td>100</td>
<td>347</td>
<td>351</td>
<td>286</td>
</tr>
<tr>
<td>8</td>
<td>100</td>
<td>504</td>
<td>510</td>
<td>424</td>
</tr>
<tr>
<td>9</td>
<td>100</td>
<td>670</td>
<td>677</td>
<td>571</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>1160</td>
<td>1170</td>
<td>999</td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>622</td>
<td>628</td>
<td>529</td>
</tr>
<tr>
<td>12</td>
<td>100</td>
<td>865</td>
<td>878</td>
<td>756</td>
</tr>
</tbody>
</table>
Table 3. The average number of attempts and comparisons performed to search for (100) patterns selected from the middle of the text.

<table>
<thead>
<tr>
<th>Pattern length</th>
<th>Number of words</th>
<th>ETSW</th>
<th>ERS-A</th>
<th>EERS-A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Attempts</td>
<td>Comparisons</td>
<td>Attempts</td>
<td>Comparisons</td>
</tr>
<tr>
<td>5</td>
<td>13965</td>
<td>11618</td>
<td>11038</td>
<td>11970</td>
</tr>
<tr>
<td>6</td>
<td>16682</td>
<td>16771</td>
<td>13536</td>
<td>14870</td>
</tr>
<tr>
<td>7</td>
<td>27267</td>
<td>26242</td>
<td>22607</td>
<td>24971</td>
</tr>
<tr>
<td>8</td>
<td>27830</td>
<td>28015</td>
<td>23385</td>
<td>25976</td>
</tr>
<tr>
<td>9</td>
<td>33929</td>
<td>34069</td>
<td>28764</td>
<td>31541</td>
</tr>
<tr>
<td>10</td>
<td>29676</td>
<td>29845</td>
<td>25471</td>
<td>28193</td>
</tr>
<tr>
<td>11</td>
<td>23195</td>
<td>23242</td>
<td>19886</td>
<td>21119</td>
</tr>
<tr>
<td>12</td>
<td>26806</td>
<td>27009</td>
<td>23484</td>
<td>26507</td>
</tr>
</tbody>
</table>

Table 4. The average number of attempts and comparisons performed to search for (100) patterns selected from the left side of the text.

<table>
<thead>
<tr>
<th>Pattern length</th>
<th>Number of words</th>
<th>ETSW</th>
<th>ERS-A</th>
<th>EERS-A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Attempts</td>
<td>Comparisons</td>
<td>Attempts</td>
<td>Comparisons</td>
</tr>
<tr>
<td>5</td>
<td>271</td>
<td>270</td>
<td>216</td>
<td>238</td>
</tr>
<tr>
<td>6</td>
<td>364</td>
<td>368</td>
<td>295</td>
<td>326</td>
</tr>
<tr>
<td>7</td>
<td>402</td>
<td>405</td>
<td>333</td>
<td>372</td>
</tr>
<tr>
<td>8</td>
<td>536</td>
<td>541</td>
<td>451</td>
<td>499</td>
</tr>
<tr>
<td>9</td>
<td>776</td>
<td>783</td>
<td>660</td>
<td>730</td>
</tr>
<tr>
<td>10</td>
<td>1579</td>
<td>1593</td>
<td>1361</td>
<td>1517</td>
</tr>
<tr>
<td>11</td>
<td>619</td>
<td>624</td>
<td>531</td>
<td>573</td>
</tr>
<tr>
<td>12</td>
<td>1667</td>
<td>1685</td>
<td>1459</td>
<td>1641</td>
</tr>
</tbody>
</table>

Table 5. The average number of attempts and comparisons for patterns with different lengths.

<table>
<thead>
<tr>
<th>Pattern length</th>
<th>Number of words</th>
<th>TSW</th>
<th>ETSW</th>
<th>RS-A</th>
<th>ERS-A</th>
<th>EERS-A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Comparisons</td>
<td>Comparisons</td>
<td>Comparisons</td>
<td>Comparisons</td>
<td>Comparisons</td>
<td>Comparisons</td>
</tr>
<tr>
<td>5</td>
<td>4535</td>
<td>4896</td>
<td>3549</td>
<td>8191</td>
<td>3880</td>
<td>2813</td>
</tr>
<tr>
<td>6</td>
<td>2896</td>
<td>8311</td>
<td>7633</td>
<td>9556</td>
<td>6750</td>
<td>6212</td>
</tr>
<tr>
<td>7</td>
<td>1988</td>
<td>10263</td>
<td>9118</td>
<td>10638</td>
<td>8506</td>
<td>7636</td>
</tr>
<tr>
<td>8</td>
<td>1167</td>
<td>11087</td>
<td>10115</td>
<td>11922</td>
<td>9319</td>
<td>8525</td>
</tr>
<tr>
<td>9</td>
<td>681</td>
<td>10538</td>
<td>9590</td>
<td>12911</td>
<td>8957</td>
<td>8171</td>
</tr>
<tr>
<td>10</td>
<td>382</td>
<td>10272</td>
<td>9339</td>
<td>12927</td>
<td>8830</td>
<td>8042</td>
</tr>
<tr>
<td>11</td>
<td>191</td>
<td>5967</td>
<td>5482</td>
<td>11672</td>
<td>5146</td>
<td>4742</td>
</tr>
<tr>
<td>12</td>
<td>69</td>
<td>7168</td>
<td>6433</td>
<td>9030</td>
<td>6286</td>
<td>5653</td>
</tr>
<tr>
<td>13</td>
<td>55</td>
<td>8673</td>
<td>7986</td>
<td>9422</td>
<td>7587</td>
<td>7004</td>
</tr>
<tr>
<td>14</td>
<td>139</td>
<td>21319</td>
<td>19535</td>
<td>17845</td>
<td>18776</td>
<td>17242</td>
</tr>
<tr>
<td>15</td>
<td>32</td>
<td>21739</td>
<td>19782</td>
<td>18318</td>
<td>19198</td>
<td>17519</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>21596</td>
<td>20092</td>
<td>23531</td>
<td>19147</td>
<td>17807</td>
</tr>
<tr>
<td>17</td>
<td>3</td>
<td>25404</td>
<td>22147</td>
<td>23119</td>
<td>22669</td>
<td>19855</td>
</tr>
</tbody>
</table>
other algorithms such as TSW, ETSW, RSA and ERS-A. For example, it took EERS-A 8042 comparisons to locate the pattern of length 10 while it took TSW, ETSW, RS-A and ERS-A 10272, 9339, 12927 and 8830 comparisons respectively for the same pattern length. These results show that EERS-A is the best algorithm compared to others.

13. Conclusions and Future Work

In this paper, we presented a new pattern matching algorithm—Enhanced ERS-A algorithm. The Enhanced ERS-A algorithm enhances the ERS-A’s process by utilizing the idea of the two sliding windows and by making comparisons with the pattern from both sides simultaneously. The comparisons done between the text and the pattern are done from both sides in parallel. This process gives the proposed algorithm a preference over the ERS-A algorithm.

The Enhanced ERS-A algorithm utilizes the idea of RS shifting algorithm to maximize the shift values. To assess the performance of the proposed algorithm, we considered ETSW and ERS-A algorithms for comparison with the proposed algorithm. The experimental results show that the Enhanced ERS-A shows better results in the number of comparisons performed.

References


