International Journal of Electrical and Computer Engineering (IJECE)
Vol.8, No.6, December 2018, pp. 4321~4333
ISSN: 2088-8708, DOI: 10.11591/ijece.v8i6.pp4321-4333

The Improved Hybrid Algorithm for the Atheer and Berry-ravindran Algorithms

Atheer Akram Abdul Razzaq[1], Nur'Aini Abdul Rashid[2], Alaa Ahmed Abbood[3], Zurinahni Zainol[4]
[1,3]Businesses Informatics College, University of Information Technology and Communications, Iraq
[2]College of Computer & Information Sciences, Princess Nourahbint Abdulrahman University, Saudi Arabia
[4]School of Computer Sciences, University Saints Malaysia, Malaysia

Article Info

Article history:

Received Aug 26, 2017
Revised Jul 12, 2018
Accepted Jul 26, 2018

Keyword:

E-Atheer algorithm
Exact string matching algorithms
Type of database

ABSTRACT

Exact String matching considers is one of the important ways in solving the basic problems in computer science. This research proposed a hybrid exact string matching algorithm called E-Atheer. This algorithm depended on good features; searching and shifting techniques in the Atheer and Berry-Ravindran algorithms, respectively. The proposed algorithm showed better performance in number of attempts and character comparisons compared to the original and recent and standard algorithms. E-Atheer algorithm used several types of databases, which are DNA, Protein, XML, Pitch, English, and Source. The best performancein the number of attempts is when the algorithm is executed using the pitch dataset. The worst performance is when it is used with DNA dataset. The best and worst databases in the number of character comparisons with the E-Atheer algorithm are the Source and DNA databases, respectively.

Corresponding Author:

Atheer Akram Abdul Razzaq,
Businesses Informatics College,
University of Information Technology and Communications,
Al-Nidhal Street, Baghdad, Iraq.
Email: athproof@uoitc.edu.iq

1. INTRODUCTION

String matching is the process of identifying all occurrences of alignments by comparing two finite-length strings [1]. String matching is among the most important problems applied in many computer science applications, such as web search engines [2], operating systems, compilers, command interpreters [3], intrusion detection systems [4], information retrieval and artificial intelligence [5]. String matching involves a matching process involving patterns and texts to identify the identical characters between them. The matching character comparisons, and the number of attempts; these factors are changeable depending on the type of algorithm used [6], [7].

Permanent challenges require the use of the most efficient string matching algorithms with increasing memory size and computer speed [8], [9]. Thus, string matching algorithms have been recently proposed to minimize these problems [10]. The hybrid string matching algorithm is the appropriate solution to mitigate disadvantages of original string matching algorithms [11]. The proposed algorithm in this paper depends on the good advantages of two exact string matching algorithms which are (Atheer and Berry-ravindran) and decreasing the disadvantages of them. All types of databases in benchmark standard are used in this research to find the suited and unsuited databases with proposed algorithm. The objective of this research overcome the weaknesses and improves the performance of exact string matching algorithms.

The original algorithms: Two original algorithms were used as referenced in this research, which are Atheer and Berry-Ravindran algorithms. Atheer algorithm is a hybrid algorithm of three algorithms which are Raita, Smith, and Karp-Rabin [3]. There are three functions preprocessing of the Atheer algorithm which are, the BM bad character (bmBc) function, the quick search bad character (qsBc) function and the hashing function. In the searching phase of the Atheer algorithm, all the comparison steps depending on the hash process which were derived from the Karp-Rabin algorithm. The comparison technique start between the hash values of the three characters (rightmost, leftmost, and middle) in the pattern with the hashing value of the three characters in the text window. If matching occurs between them, then one by one these three characters of the text window versus the three characters of the pattern will be compared. When matching occurs, then comparison starts in the remaining characters, but without comparing the middle character again. When a matching or mismatching occurs, the shifting of the pattern would depend on the maximum value between the m value in the bmBc table and on the m+1 value in the qsBc table.

The Berry-ravindran algorithm is a hybrid approach and is characterized by left-right character comparisons [12]. This algorithm is a hybrid of the Zhu-takaoka and Quick-search algorithms and has two phases, namely, preprocessing and searching. The preprocessing phase of this algorithm depends on the Berry-ravindran bad character (brBc) function. The searching phase of the Berry-Ravindran algorithm has left-right character movements. This algorithm depends on the shifting operation of the next two characters in the text window, which depends on the m+1 and m+2 of the text window. The shifting value is obtained from the brBc table in the preprocessing phase. The comparison process starts from the leftmost character in the pattern with the leftmost character in the text window. If a match is found, the comparison will continue to another character until all characters are matched. When a matching or mismatching occurs, the shifting depends on the next two characters of the text window (m+1 and m+2) and the obtained value from the brBc table in the preprocessing phase.

2. PROPOSED E-ATHEER ALGORITHM

The proposed E-Atheer algorithm consists of two phases, namely, the preprocessing phase and searching phase.

2.1. Preprocessing Phase

The preprocessing phase contains the techniques selected from the Atheer and Berry-Ravindran algorithms. These techniques are regulated in functions to obtain the exact string matching of the E-Atheer algorithm. These functions are presented as follows:

a) Boyer-moore Bad Character (Bmbc) Function

The technique selected from this function is similar to that in the preprocessing phase of the Atheer algorithm. The main purpose of using the bmBc table in this function is to determine and choose the best shifting for each character in the matching operation as shown in Lines 21–26, Figure 1. The form of the bmBc function is defined by the equation below.

$$\left\{\begin{array}{l} \min\{i: 1 \le i < m-1 \text{ and } x[m-1-i] = c\} \\ \text{if } c \text{ occurs in } x, \\ bmBc[c] \\ \\ m \text{ otherwise} \end{array}\right\} \qquad (1)$$

b) Berry-ravindran Bad Character (Brbc) Function

The technique in this function is selected from the Berry-Ravindran algorithm. The main purpose of using the brBc table in this function is to determine and choose the maximum value in the shifting process as shown in Lines 5-19, Figure 1. The form of the brBc function is defined by the equation below. The main text format consists of a flat left-right.

$$\left\{\begin{array}{ll} 1 & \text{if } P[m-1] = u, \\ m - i + 1 & \text{if } P[i]\,P[i+1] = uv, \\ m + 1 & \text{if } P[0] = v, \\ m + 2 & \text{otherwise.} \\ brBc[u, v] = \mathit{min} & \end{array}\right\} \qquad (2)$$

c) Hashing Function

This function is derived from the Atheer algorithm and the hashing technique of E-Atheer algorithm, which is similar to the hashing technique of the Atheer algorithm. The proposed hybrid algorithm uses the (Fh) and (Fhw) symbols for first hashing step in pattern and first hashing step in the text window, respectively. It uses the (Sh) symbol for second hashing and (Th) for third hashing steps in the pattern as shown in Lines 28–34; Figure 1. The hashing value is calculated using the following equations:

First hashing step**:** $(wF\ [0, m/2, m\text{-}1]) = (wF\ [0] \times 2u\text{-}1 + wF\ [m/2] \times 2u\text{-}2 + wF\ [m\ \text{-}1] \times 20) \bmod q$ (3)

Second hashing step: $(wS\ [1\ldots\ m/2\text{-}1]) = (wS\ [1] \times 2u\text{-}1 + wS\ [2] \times 2u\text{-}2 + \ldots + wS\ [m/2\text{-}1] \times 20) \bmod q$ (4)

Third hashing step: $(wT\ [m/2+1 \ldots\ m\ \text{-}2]) = (wT\ [m/2+1] \times 2u\text{-}1 + wT\ [m/2+2] \times 2u\text{-}2 + \ldots + wT\ [m\text{-}2] \times 20) \bmod q$ (5)

```
1.  Algorithm E-Atheer(X [0 …..m−1]
2.  //Input: Pattern X
3.  //Output: Shift tables of (bmBc), (brBc) and compute the hush values.
4.  //prebrBc (preprocessing Berry-Ravindran bad-character function)
5.  brBc[ASIZE][ASIZE]  //2D array to keep shift values
6.      For k← 0 to ASIZE Do
7.          For j ← 0  to ASIZE Do
8.  brBc[k][j] ←m+2
9.          End For
10.     End For
11.     For k←0 to ASIZE Do
12. brBc[k][x[0]]← m +1
13.     End For
14.     For i ←0 to m−2 Do
15. brBc[x[i]][x[i+1]] ←m − i
16.     End For
17.     For k ←0 to ASIZE Do
18. brBc[x[m−1]][k] ←1
19.     End For
20.     //prebmBc (preprocessingBoyer-Moore bad-character function)
21.     For j ←0 to size of alphabet Do
22. bmBc[j] ←m
23.     End For
24.   For j ←0 to m−2 Do
25. bmBc [X[j]]← m− j−1
26.   End For
27.   // Compute the hush values h = d^S−1 mod q
28.  For i← w  to S−1 Do
29. hy←(hy<<1)+y[i]
30.  End For
31. firstCh←x[0], secondCh ← x+1, middleCh ← x[m/2], lastCh←[m−1]
32.  // Hash values of all steps in pattern and the first three characters in text window
33. fhx← (fhx<<1) + firstCh,  fhx ← (fhx<<1) + middleCh,  fhx←(fhx<<1) + lastCh
34. fhy← (fhy<<1) + y[0],   fhy ← (fhy<<1) + y[m/2],  fhy ← (fhy<<1) + y[m−1]
```

Figure 1. Preprocessing phase in the E-Atheer algorithm

2.2. Searching Phase

The searching phase technique in the E-Atheer algorithm depends on the searching phase techniques of the Atheer and Berry-Ravindran algorithms and on some of the modulations obtained during the matching operation. In the first step, the hash values of the three characters in pattern (Fh) are compared with the hash values of the three characters (Fhw) in the text window. If a match is obtained between the hashing values, the remaining three characters in the text window and the remaining three characters in the pattern will be compared. If a match is obtained between these characters, the second step will be conducted as shown in Line 11; Figure 2. However, if a mismatch is obtained in the hashing comparison or in the character comparison, the new shift of this algorithm will depend on the maximum value of m from the bmBc table and the (m+1 and m+2) values from the brBc table as shown in Line 24; Figure 2. The m refers to the last character in the text window, the m+1 is the first character after the text window, and m+2 is the second character after the text window. The rehash function is then used to calculate the three characters of the new text window after the shift as shown in Line 26; Figure 2.

```
1. Algorithm E-Atheer (X [0 …..m−1], Y [0…….n−1])
2.  //Input: Pattern X, Text Y
3.  //Output: number of attempts and number of character comparisons of pattern with text
4.  If (m%2 == 0) Then
5.      par ← 1
6.  End If
7.  j ←0
8.  While j <= n − m Do
9.      c ← y[j + m − 1]
10.     // Comparing the Fh and Fhw
11.     If (fhx == fhy&&lastCh == c &&firstCh == y[j]&&middleCh ==y[j + m/2]) Then
12. shfy← gethy(j + 1, j + m/2, y)  //calculate the hash of  (Shw)
13.         // Comparing the Sh and Shw
14.         If (shfx == shfy&& match(x + 1, m/2−1, y, j + 1, &temp) == 1) Then
15. shly← gethy(j+m/2+1, j + m−1, y)  // calculate the hash of  (Thw)
16.             // Comparing the Th and Thw
17.             If(shlx == shly&& match(x + m/2 + 1, m/2−1-par, y, j + m/2+ 1, &temp)  == )Then
18.                 Count   //The first occurrence of the pattern in the text
19. EndIf
20. EndIf
21. EndIf
22.     Output the first attempt and character comparisons
23.     //shifting//
24.     j +=max(brBc[y [j + m]][y[j+m+1]],bmBc[y[j + m−1]])
25.    // Rehash operation for the text window
26. fhy← 0,fhy ← (fhy<<1) + y[j],fhy ← (fhy<<1) + y[j+m/2],fhy← (fhy<<1) + y[j+m−1]
27. End While
```

Figure 2. Searching phase in the E-Atheer algorithm

In the second step, when a match is obtained in the first step, the hashing from the second to middle -1 characters in the text window (denoted by Shw) is calculated and is compared with the hashing characters (Sh) in the pattern. If a match is obtained, the comparison of characters between will continue (Lines 12 and 14, as shown in Figure 2. If another match is obtained between the characters, the third step will commence. If a mismatch is obtained between the characters, the shift will depend on the same technique mentioned in the previous step as shown in Line 24; Figure 2 and the rehash function will be used. In the third step, when a match is obtained in the second step, the hashing from the middle +1 to last −1 characters in the text window (denoted by (Thw)) is calculated and is compared with the hashing characters (Th) in the pattern. If a match is obtained, the comparison of the characters between the characters will continue as shown in Lines 15 and 17; Figure 2. If a match or a mismatch is obtained between the characters, the shift will depend on the same technique mentioned on the previous steps as shown in Line 24; Figure 2 and the last step (i.e., the rehash function) will commence.

3. PROPOSED ALGORITHM ANALYSIS

Preprocessing of the proposed algorithm has three functions which are brBc, bmBc and hash function. The time complexity of brBc function is denoted as $O(m+\sigma^2)$, the bmBc function is denoted as $O(m+\sigma)$ and hash function is denoted as $O(m)$. The space and time complexity of the preprocessing phase of the proposed algorithm is denoted as $O(m+\sigma^2)$. The time complexity of the searching phase explains in the following section. Lemma.3.1 The time complexity of the searching phase is $O(n/(m+2))$ in the best case. Proof. In each attempt, if any character does not happen in the pattern during the matching process, then the shifting process will depend on maximum value between m from bmBc and (m+1 and m+2) from brBc functions that calculated in the preprocessing phase. The best case occurs when all characters in the pattern totally different than those in the text window, then the shifting will depend on m+2 and the time complexity will be $O(n/(m+2))$.

For example: Text: yyyyyyyyyyyyyyyy
Pattern: xxxx

Lemma.3.2 The time complexity of the searching phase is $O(n \times m)$ in the worst case.

Proof. In the matching process every character in the text does not occur more than m times and all the character comparisons for n characters of the text cannot be greater than $n \times m$. The worst case happens if all the characters in the pattern are same with those in the text window in every attempt. Then the shifting

will be one and the time complexity will be O (n × m).
For example: Text: yyyyyyyyyyyyyyyy
Pattern: yyyy

In this algorithm cannot accurately determine the average time complexity because of its dependence on the alphabet size of characters and the possibility of the appearance of each character individually in the text.

4. EXPERIMENTAL DESIGN AND EVALUATION OF STUDY OUTCOMES

The proposed algorithm design depended on selecting the good features of original algorithms, which are the hash and bmBc functions from Atheer algorithm and brBc function from Berry-Ravindran algorithm. The proposed algorithm used all types of benchmark standard databases and the results of E-Atheer compared to results of the original and recent and standard algorithms.

4.1. Databases

This study investigates the differences in the performance and properties of several exact string matching algorithms when different types of databases are used (200 MB data size). The benchmark standard of databases deals with common types of data, such as DNA, Protein, XML Pitch, English text, and Source code. These datasets were downloaded from the Pizza & Chili Corpus Web site (http://pizzachili.dcc.uchile.cl/ (Pizza Chili Corpus). Two pattern lengths were used in this study: the short pattern length, which ranged from 4 characters to 28 characters, and the long pattern length (length power of two), which ranged from $2^{\wedge 5}$ characters to $2^{\wedge 10}$ characters [13], [14]. The DNA data sequence is composed of four nucleotides, namely, Adenine, Guanine, Cytosine, and Thiamin, and these types are encoded as A, G, C, and T respectively.

The Gutenberg project is included in this database [15], [16]. The Proteins are necessary to the structure and function of the cells of an organism. The Protein data sequence is composed of 20 amino acids arranged in a linear series and encoded using uppercase letters [17], [18]. The databases were obtained from the Swiss-Prot database. The XML structure database contains the bibliographic information of computer sciences. The Pitch (Midi Pitch values) database contains tuning data used in digital music [19], [20]. The English text database uses all the characters in the English alphabet. The Gutenberg project has established this database [15]. The Source program code database is composed of all the characters used in the C and Java languages [21].

4.2. Implementation and Environment

This experiment was conducted using the Biruni cluster in the School of Computer Sciences at USM (biruni.cs.usm.my). The operating system used was Ubuntu Linux 10.04 and the compiler used was GCC v4.4.3. This study showed that the hybrid algorithm was “best in performance” as the result of the hybrid algorithm was better than those of the original algorithms. The tables of evaluation for each hybrid algorithm were arranged based on the best result and followed by the other algorithms. The algorithms were then ranked as “first,” “second,” or “third,” In the evaluation the performance of the hybrid algorithm in various types of databases, the results are regarded as “best” when the hybrid algorithm performed better in specific databases compared with the other algorithms, whereas “worst” refers to the poorest performance of the hybrid algorithm for that database. When the hybrid or other algorithms obtained the best performance in all types of databases, then “all databases” isused, whereas when the hybrid or other algorithms obtained the best performance in almost all databases, “most databases” is used. To clarify the results in the figures in number of attempts, the proposed hybrid algorithms show only (10000) display units compared with the original algorithm. Compared with the recent and standard algorithms, the proposed algorithm has a logarithmic scale and base of (10), display units of (10000), and minimum number of (100000). In number of character comparisons, the proposed hybrid algorithm has a logarithmic scale and base of (10) and display units of (10000) compared with the original, standard, and recent algorithms.

5. RESULTS AND DISCUSION

The results of E-Atheer algorithm compared with those of the original algorithms in first step and then with those of the recent and standard algorithms in second step. Number of attempts and number of character comparisons considered the main factors that used in this study. The databases used in this study are DNA, Protein, XML, Pitch, English, and Source. The size of data is 200 MB. Two pattern lengths were used, which are short (4 to 28) and long that depends on the length power of two ($2^{\wedge 5}$ to $2^{\wedge 10}$).

5.1. Evaluation and Results Analysis of the E-Atheer Algorithm and the Original

The E-Atheer algorithm obtains the best results compared with the Berry-Ravindran and Atheer algorithms in both short and long patterns. The Pitch database shows the best results in number of attempts when using short and long patterns, whereas the DNA database show the worst results as shown in Figures 3 and 4. The E-Atheer algorithm obtains the best results compared with the Atheer and Berry-ravindran algorithms in both short and long patterns. The Source database shows the best results in number of character comparisons when using short and long patterns, whereas the DNA database shows the worst results as shown in Figures 5 and 6.

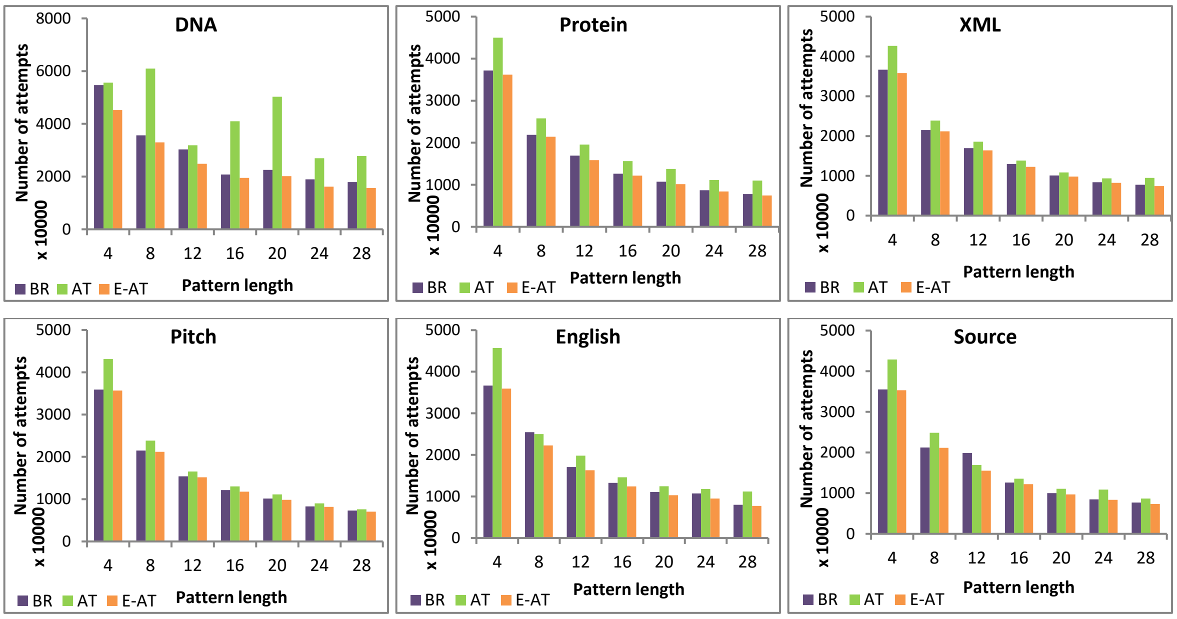

Figure 3. Number of attempts for E-Atheer compared with the original algorithms when using a short pattern and a 200 MB data size

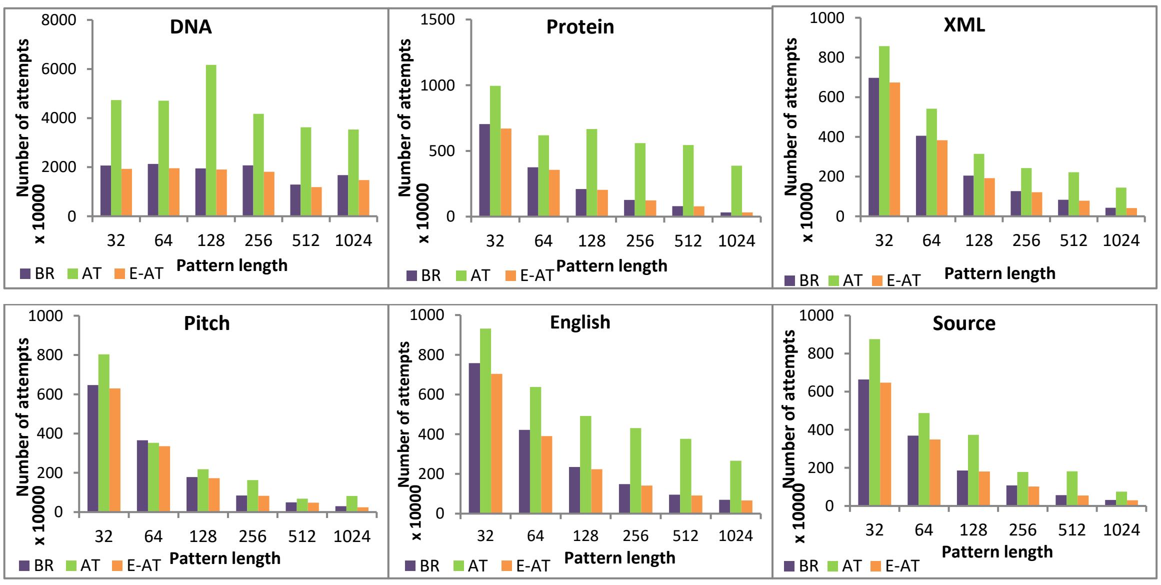

Figure 4. Number of attempts for E-Atheer compared with the original algorithms when using a long pattern and a 200 MB data size

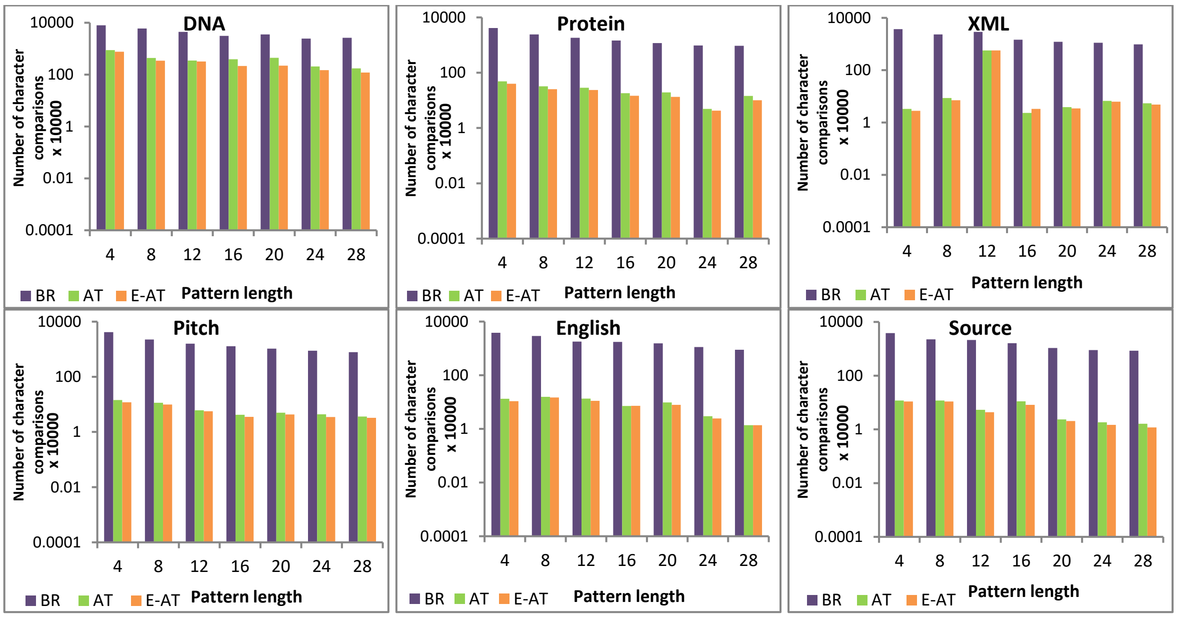

Figure 5. Number of character comparisons for E-Atheer compared with the original algorithms when using a short pattern and a 200 MB data size

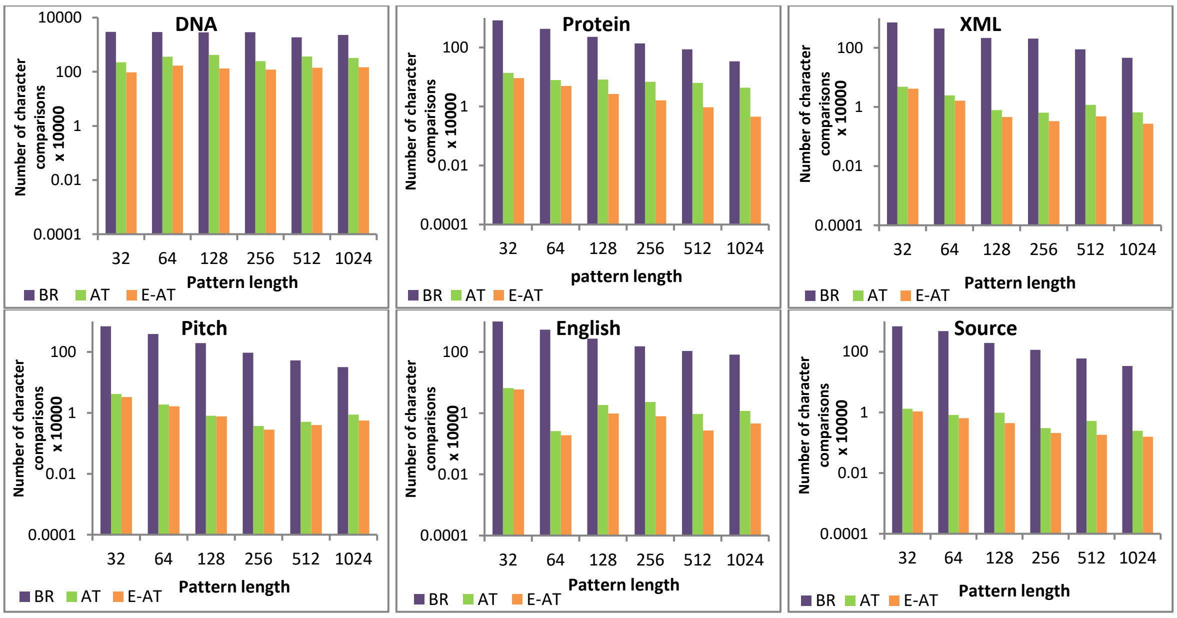

Figure 6. Number of character comparisons for E-Atheer compared with the original algorithms when using a long pattern and a 200 MB data size

5.2. Evaluation and Results Analysis of the E-Atheer Algorithm and Recent and Standard Algorithms

The E-Atheer algorithm considered the best algorithm in all types of databases when using short pattern except when using DNA it was the second best. The Maximum shift algorithm is the best algorithm in all databases when using long pattern and followed by the E-Atheer algorithm except when using Pitch database it is the best with E-Atheer. The Two-way algorithm is the worst algorithm in short and long pattern lengths as shown in Figures 7 and 8.

The E-Atheer algorithm considered the best algorithm in all databases when using short pattern, while AKRAM is the best algorithm in all databases and E-Atheer is the second best algorithm using long pattern length in all databases except XML the E-Atheer and AKRAM are the best. The Two-way is the worst algorithm in short and long pattern lengths as shown in Figures 9 and 10.

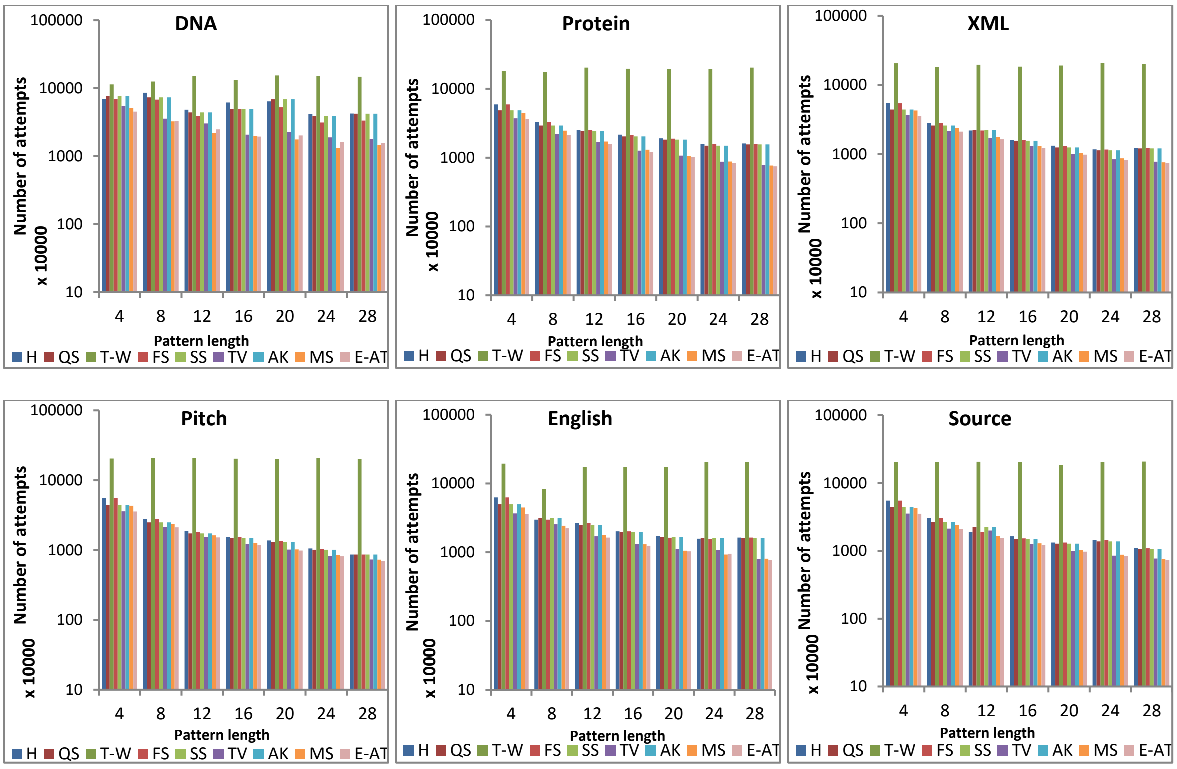

Figure 7. Number of attempts for E-Atheer compared with recent and standard algorithms when using a short pattern and a 200 MB data size

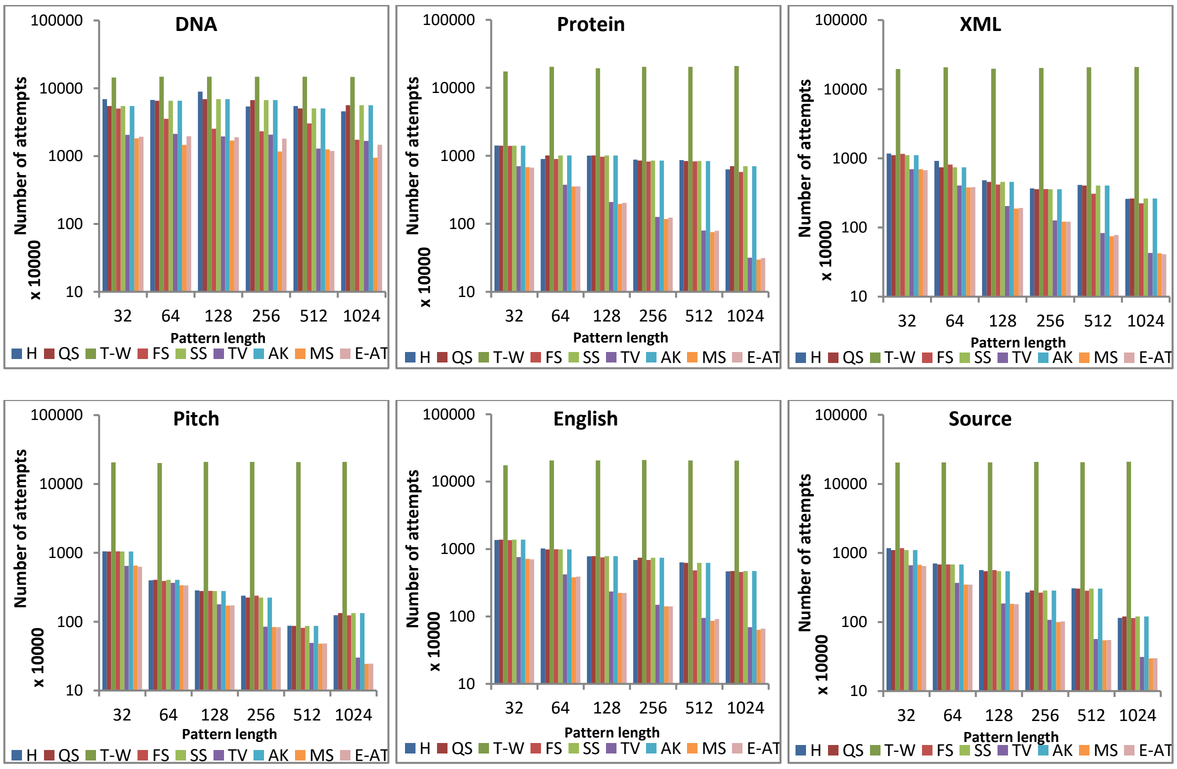

Figure 8. Number of attempts for E-Atheer compared with recent and standard algorithms when using a long pattern and a 200 MB data size

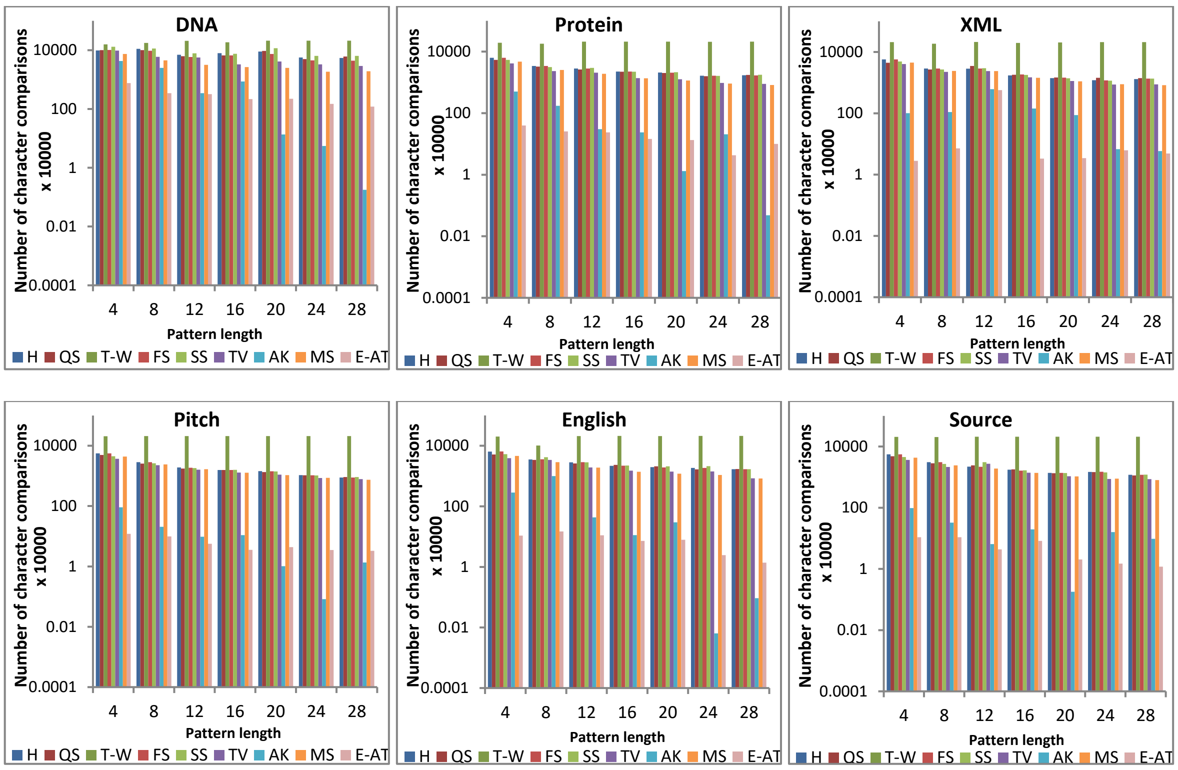

Figure 9. Number of character comparisons for E-Atheer against recent and standard algorithms when using a short pattern and a 200 MB data size

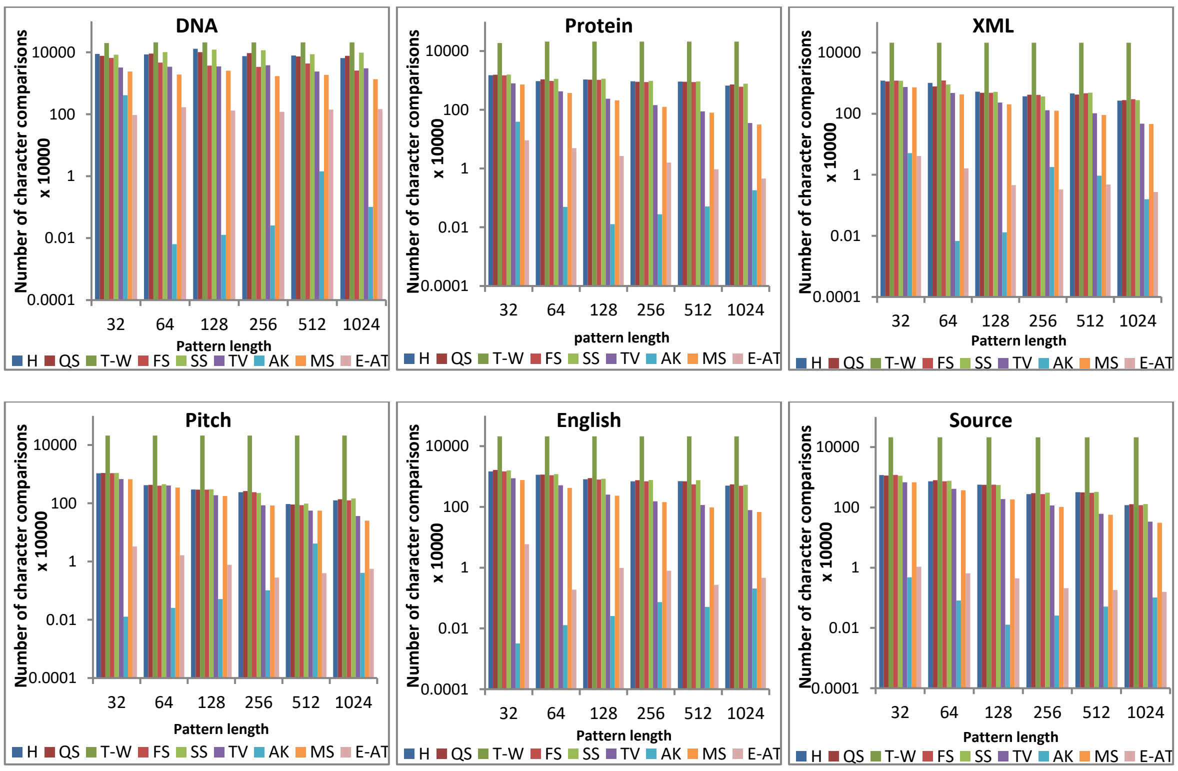

Figure 10. Number of character comparisons for E-Atheer against recent and standard algorithms when using a long pattern and a 200 MB data size

5.3. Evaluation of the E-Atheer Algorithm Compared with the Original Algorithms

The performance results of the E-Atheer algorithm and the original algorithms are compared in terms of the number of attempts and the number of character comparisons in both short and long patterns with different data types and sizes. Table 1 shows comparison of the results of the e-atheer and original algorithms.

Table 1. Comparison of the Results of the e-atheer and Original Algorithms

Algorithms	Data size (200MB)	
	Short	Long
Best performance in Number of attempts		
Berry-Ravindran	Second	Second
Atheer	Third	Third
E-Atheer	First	First
Best performance in Number of character comparisons		
Berry-Ravindran	Third	Third
Atheer	Second	Second
E-Atheer	First	First

The E-Atheer algorithm obtains the best results in terms of the number of attempts made because it depends on the best shifting function from the maximum value of brBc and bmBc. The E-Atheer algorithm obtains the lowest number of character comparisons because it relies on the hash function, thus simplifying the character comparison between patterns and texts [22]. The best shifting functions also help reduce the number of character comparisons as shown in Table 1. Table 2 shows performance of the e-atheer algorithm in different database types.

Table 2. Performance of the e-atheer Algorithm in Different Database Types

Performance	Databases		
	Data size	200 MB	
	Pattern length	Short	Long
Attempts	Best	Pitch	Pitch
	Worst	DNA	DNA
Character comparisons	Best	Source	Source
	Worst	DNA	DNA

The E-Atheer algorithm obtains the fewest number of attempts in the Pitch database because this algorithm depends on two good functions, namely, hash and bmBc, in the Atheer algorithm [3]; these functions are considered efficient when employed in the Pitch database [23]. Pitch data contain a high percentage of numbers because the data are encoded as MIDI pitch numbers in computer applications [24], [25]. The hash function also uses integer numbers and the bmBc function, which is considered a good shifting function that helps reduce the number of attempts.

The E-Atheer algorithm obtains the lowest number of character comparisons in the Source code because it relies on the efficiency of the Atheer technique. The Source database also benefits from this technique. The two algorithms use the hash function in databases with large alphabet sizes to produce large hash values, thus reducing the probability of character comparison. The E-Atheer and Atheer algorithms show the highest number of attempts and character comparisons in the DNA database as shown in Table 2.

5.4. Evaluation of the E-Atheer Algorithm Compared With Recent and Standard Algorithms

The performance results of the E-Atheer algorithm and the recent and standard algorithms were compared in terms of the number of attempts and character comparisons using short and long patterns, with different data types and sizes. The standard and recent algorithms employed in this study are Horspool, Quick-search, Two-way, Fast search, SSABS, TVSBS, AKRAM, and Maximum shift. Table 3 shows comparison of the results between the e-atheer algorithm and recent and standard algorithms.

The E-Atheer algorithm obtains the fewest number of attempts when using short patterns because this algorithm depends on the efficient shifting functions (bmBc and brBc) of the Atheer algorithm. The Maximum shift algorithm shows the fewest number of attempts because this algorithm relies on the efficient functions of (ztBc) and (qsBc) in long patterns [26]. The E-Atheer algorithm obtains the lowest number of character comparisons when using short patterns because this algorithm depends on the useful technique of the Atheer algorithm in comparing characters. If a mismatch is obtained in the second step, the loss will only involve three characters because the first step depends on three characters only [3]. The AKRAM algorithm also obtains the lowest number of character comparisons in long patterns because the high hash value in long patterns reduces the mismatch probability [27]. The Two-way algorithm shows the worst results in terms of the number of attempts and character comparisons because this algorithm depends on the factorization technique [8] as shown in Table 3. Table 4 shows ranking of the e-atheer algorithm in different data types.

Table 3. Comparison of the Results between the e-atheer Algorithm and recent and Standard Algorithms

Algorithms	Data size (200MB)	
	Short	Long
	Number of attempts	
Best algorithm	E-Atheer (most databases)	Maximum shift (all databases)
Worst algorithm	Two-way (all databases)	Two-way (all databases)
	Number of character comparisons	
Best algorithm	E-Atheer (all databases)	AKRAM (all databases)
Worst algorithm	Two-way (all databases)	Two-way (all databases)

The E-Atheer algorithm ranks first in most data types and sizes when short patterns are used in terms of the number of attempts performed. This algorithm ranks second in most databases when using long patterns. For the number of character comparisons, the E-Atheer algorithm ranks first in all databases with different sizes when short patterns are used. The E-Atheer algorithm ranks second in most databases when long patterns are used as shown in Table 4.

Table 4. Ranking of the e-atheer Algorithm in Different Data Types

Databases	Pattern length	Ranking of E-Atheer algorithm Data size 200 MB
Attempts	Short	First in all databases (but second in DNA)
	Long	First in Pitch, Second in DNA, Protein, XML, English & Source
Character comparisons	Short	First in all databases
	Long	First in XML, Second in DNA, Protein, Pitch, English & Source

6. CONCLUSION

The E-Atheer algorithm obtains the best results in terms of the number of attempts compared with the original algorithms when short and long pattern lengths are used. The algorithm rank first in short patterns compared with the recent and standard algorithms and rank second in some of data types when long patterns are used. For the number of character comparisons, the proposed algorithm show the best results compared with the original algorithms in short and long pattern lengths. The improved algorithm also performs better than the recent and standard algorithms in short pattern lengths, while it ranks second in long patterns. The Pitch database shows the best performance in the number of attempts with the proposed algorithm, whereas the DNA database performs the worst. The best and worst databases in the number of character comparisons with the E-Atheer algorithm are the Source and DNA databases, respectively.

REFERENCES

[1] S. Faro and T. Lecroq, “The Exact Online String Matching Problem: a Review of the Most Recent Results”, *ACM computing survey*, vol. 45(2), pp.1-42, 2013.
[2] B. Leonardo and S. Hansun, "Text Documents Plagiarism Detection using Rabin-Karp and Jaro-Winkler Distance Algorithms", Indonesian Journal of Electrical Engineering and Computer Science (ijeecs), Vol. 5(2), pp. 462 - 471,2017.
[3] A.A. Abdul Razzaq, *et al.*, “A New Efficient Hybrid Exact String Matching Algorithm and Its Applications”, *Life Science Journal (Life Sci J)*, vol. 11(10), pp.474-488, 2014.
[4] A.A. Hasan and N.A. Rashid, “Hybrid Exact String Matching Algorithm for Intrusion Detection System, *Proceedings of the Second International Conference on Communications and Information Technology, ICCIT 2012, 2012.*

[5] A.A. Abdul Razzaq, *et al.*, “Influenced Factors on Computation Among Quick Search, Two-Way and Karp-Rabin Algorithms”, *Proceeding of the 3rd International Conference on Informatics and Technology, Informatics '09*, pp. 100-106, 2009.
[6] T. Lecroq, “Experimental Results on String Matching Algorithms”, *Software-Practice and Experience*, vol. l (25), pp. 727–765, 1995.
[7] I. Hussain, *et al*., “Improved-Bidirectional Exact Pattern Matching”, *International Journal of Scientific & Engineering Research*, vol. 4 (5), pp.659-663, 2013.
[8] C. Charras and T. Lecroq, “Handbook of Exact String Matching algorithms”, *King’s College Publications*, 2004.
[9] V. Radhakrishna, *et al*., “A Two Way Pattern Matching Algorithm Using Sliding Patterns”, *3rd International Conference on Advanced Computer Theory and Engineering, ICACTE*,2010.
[10] M. Zubair, et al., “Improved Text Scanning Approach for Exact String matching”, International Conference on Information and Emerging Technologie, ICIET. pp. 1-5, 2010.
[11] M.A S. Naser, et al., “Quick-Skip Search Hybrid Algorithm for the Exact String Matching Problem”, *International Journal of Computer Theory and Engineering*, vol. 4(2), pp.259-265, 2012.
[12] T. Berry and S.A. Ravindran, “fast string matching algorithm and experimental results”, *In Proceedings of the Prague Stringology Club Workshop`99, J. Holub and M. Simáneked, Collaborative, Report DC-99-05*, pp. 1-17, 1999.
[13] Y. Huang, *et al*., “A Fast Improved Pattern Matching Algorithm for Biological Sequences”, *International Symposium on Computational Intelligence and Design, ISCID '08, IEEE, pp. 8-12*, 2008.
[14] G. Cai, *et al*., “A Fast Hybrid Pattern Matching Algorithm for Biological Sequences”, *Proceedings of 2nd International Conference on Biomedical Engineering and Informatics, BMEI '09*, pp. 1-5, 2009.
[15] J. Karkkainen and C.N. Joong, “Faster Filters for Approximate String Matching”, *Proceedings of the Nine Workshop on Algorithm Engineering and Experiments, ALENEX*, pp. 84-90,2009.
[16] K. Saputra, et al., "Fuzzy-based Spectral Alignment for Correcting DNA Sequence from Next Generation Sequencer", TELKOMNIKA, Vol. 14(2), pp. 707-714, 2016.
[17] A.F. Klaib and H. Osborne, “BRQS Matching Algorithm for Searching Protein Sequence Databases”, *Proceedings of the 2009 International Conference on Future Computer and Communication, ICFCC '09 IEEE*, pp. 223-226, 2009.
[18] V. Kurt, “Protein Structure Prediction using Decision Lists”, *M.Sc. Thesis, Dept, Computational Sciences and Engineering, Koç University, İstanbul, Turkey*, 2005.
[19] E. Chew and Y.C. Chen, “Mapping Midi to the Spiral Array: Disambiguating Pitch Spellings”, In H. K. Bhargava and Nong Ye, eds. Computational Modeling and Problem Solving in the Networked World, *Proceedings of the 8th INFORMS Computer Society Conference, ICS2003*, pp. 1-17, 2003.
[20] J. Doherty, et al., "Streaming Audio Using MPEG–7 Audio Spectrum Envelope to Enable Self-similarity within Polyphonic Audio", TELKOMNIKA, Vol. 15(1), pp. 190-202, 2017.
[21] P. Ferragina and J. Fischer, “Suffix Arrays on Words. In: Bin Ma, Kaizhong Zhang (eds.)”, *Proceedings of the 18th Annual Symposium on Combinatorial Pattern Matching, CPM’07*, pp. 328-339, 2007.
[22] R.A. Deighton, “Using Rabin-Karp fingerprints and LevelDB for faster searches”, *M.Sc. Thesis, Dept, Department of Computer Science, University of Ontario Institute of Technology (UOIT), Oshawa, Canada*, 2012.
[23] R. Nidadavolu, “Content-based Retrieval of Music Using Monophonic Queries on a Database of Polyphonic*”, MIDI Information. ProQuest information and learning company*, 2008.
[24] E.Cambouropoulos, “Automatic pitch spelling: From numbers to sharps and flats”, *In VIII Brazilian Symposium on Computer Music SBC&M*, 2001.
[25] A. Honingh, “Pitch spelling: Investigating reductions of the search space”, *Proceedings SMC'07, 4th Sound and Music Computing Conference,* 2007.
[26] H.A. Kadhim, “New Sequential and Gpu-Based Hybrid String Matching Algorithms”, *M.Sc. Thesis, Dept, Computer Science School, University Science Malaysia (USM), Penang, Malaysia*, 2012.
[27] A.A. Abdul Rozaq, “Fast Hybrid String Matching Algorithm Using Message Passing Programming Model”, *M.Sc. Thesis, Dept, Computer Science School, University Science Malaysia (USM), Penang, Malaysia,* 2009.

BIOGRAPHIES OF AUTHORS

Dr. Atheer Akram Abdul Razzaq was born in Baghdad, Iraq. He got his bachelor from Al Mustansiriya University, Iraq in 2006. He got his M.Sc. from Universiti Sains Malaysia in 2009. He got his Ph.D in High performance computing (Parallel tools and applications) in 2014, School of Computer Sciences, Universiti Sains Malaysia. He is currently a senior lecturer at the Businesses Informatics College, University of Information Technology and Communication, Baghdad, Iraq. His main research area is High Performance Computing. His research interests are in exact string matching algorithms, parallel and distributed processing, network security, and data mining. Dr. Abdul Razzaq is Deputy Dean for Academic and Scientific Affairs in Businesses Informatics College. He has published numerous papers in string matching, parallel and distributed processing, network security, and genomic information Processing.

Associate Prof. Dr. Nur'Aini Abdul Rashid was born in Singapore. She got her bachelor from Mississippi State, U.S.A. She got her M.Sc. and PhD from Universiti Sains Malaysia. She was with the School of Computer Sciences, University Saints Malaysia for more 25 years. Currently she is an Associate Professor at Department of Computer Sciences in Princess Nourahbint Abdulrahman University, Kingdom of Saudi Arabia. Her main research area is High Performance Computational Biology. Her interest now is applying the parallel techniques to genomic information retrieval and analysis, and biological big data mining. The method that she is investigating is on shared memory platforms. The concentration is on parallel sequence comparison and parallel graph-based clustering algorithms. Dr. Abdul Rashid was member in Association of Computing Machinery (ACM) from 2002 to 2010, International Society of Computational Biology from 2008 to 2009and MASBIC, Malaysian Bioinformatics from 2007 to2010. She has published more than 70 refereed paper in the field of Bioinformatics, String matching and Information retrieval.

Dr. Alaa Ahmed Abbood Al-shammariwas born in Baghdad, Iraq. He got his bachelor from Al Mustansiriya University, Iraq in 2007. He got his M.Sc. from Hamdard University, New Delhi, Indai. He got his Ph.D in image processing (fingerprint classifications) in 2014, faculty of computing, Universiti Teknolgi Malaysia (UTM), Johor Bahru, Malaysia. He is currently a senior lecturer at the Businesses Informatics College, University of Information Technology and Communication, Baghdad, Iraq. His main research area is Image Processing. His research interests are in biometrics, human identifications, medical image enhancement and segmentations, and data hiding. Dr. Alaa Al-Shammari is Head of Business Information Technology Department in Businesses Informatics College. He has published numerous papers in string biometrics system, medical image and steganography.

Associate Prof. Dr. Zurinahni Zainol was born in Kedah, Malaysia. She has Bachelor (Hons) Computer Science from UITM-UKM, Malaysia, MSc (Artificial Intelligence) from Universiti Sains Malaysia. She got her PhD from University of Hull, UK. She is an Associate Professor at the School of Computer Sciences, University Saints Malaysia, Malaysia. Her research interests are in the Data Modeling, XML Database Design, Information Retrieval in Multidisciplinary such as in Education and Computational Biology. Dr. Zainol has published numerous papers in XML database design, database integration for bioinformatics application and data modeling and information retrieval in Education.