Computer Engineering and Intelligent Systems
ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online)
Vol.7, No.6, 2016

Advanced Searching Algorithms and its Behavior on Text Structures

Abid Thyab Al Ajeeli,
Higher Education Committee, C.O.R, Iraq

Abstract
This research investigates the behavior of the Boyer-Moore-Horspool (BMH) and the Boyer-Moore-Raita (BMR) string-matching algorithms using multilingual texts. The performance is computed based on searching for patterns in master strings. Experiments are conducted using a number of pattern lengths with many experiments repetition. The experimental results show that on average the number of comparisons per character passed in the case of the BMR is less than the number encountered by the BMH variant. The improvement is due to properties of the text structures. These experiments may lead to more theoretical and practical studies to develop new variants of algorithms. Using multilingual text structures provide more insight into the theory and structure of algorithms as multilingual text structures have different set of characters and dependencies, and the character properties have different type of structures. Since many applications of today depend on searching algorithms, therefore researchers need to explore every possibility that lead to improving the efficiency of searching and matching mechanisms. The time performance of exact string pattern matching can be greatly improved if an efficient algorithm is used. Considering, for example, the growing amount of text handled in the electronic patient records, it is worth and essential, in these cases and others, to searching for an efficient algorithm to deal with such huge items of information.
Keywords: Matching, Boyer-Moore, Raita algorithm, Searching, multilingual

1. Introduction

There has been much research in recent years on string searching methods. String searching is one of the most frequent operations encountered in many of today's applications such as word processing, natural language processing, computerized library systems, virus scanning, computer security, signal processing, search engines, satellite transmissions, and in genetic sequences. The problem of string searching is considered as finding the position (s) of occurrence (s) of a given pattern P of length m characters in a text string T of length n where m <= n. In practical situations n is very large compared with m.
Throughout the paper we adapt the following notations (Knuth et al. 1977):

$T = t_1t_2\ldots t_n$	{ the master text string to be searched. Its length is equal to n}
$P = p_1p_2\ldots p_m$	{ the pattern to search for in text string T. Its length is equal to m }
n	{ the master string length}
m	{ the pattern length}
t_i	{ the ith character in the master text string }
p_j	{ the jth character in the pattern}
Σ	{ the alphabet}
σ	{ the alphabet size}

One of the fastest known algorithms is that of the Boyer and Moore (Boyer & Moore 1977). Many researchers have studied it extensively. In this paper, we investigate the performance of the fastest two variations of the Boyer-Moore (BM) method (Boyer-Moore-Horspool), BMH, and Boyer-Moore-Raita, (BMR). The algorithms are applied to text/patterns from Arabic alphabet in order to identify whether the performance is due to properties of the texts being searched or to any other guarding factors.

The motivation for conducting this study was the articles published by Raita (1992) and smith (1994) who claim inconsistent conclusions when they applied the algorithms on English texts. In our experiment, Arabic text is used for the investigation of the performance of the BMH and BMR algorithms. This is accomplished by transforming the preprocessing character set of the algorithms into Arabic character set.

In order to acquaint readers with a brief description of Arabic structure, we illustrate below some basic characteristics. The structure of the Arabic language is different from other English and European language structures. For example the Arabic script is written from right to left and it is an inflectional language. Arabic language is distinguished by its high syntactical flexibility (Xu et al. 2002). This flexibility includes: the omission of some prepositional phrases associated with verbs; the possibility of using several prepositions with the same verb while preserving the meaning (Young-Suk et al. 2003).

Arabic rules allow a great deal of freedom in the ordering of words in a sentence. A sentence in Arabic may consist of one word only, which provides a complete meaning such as **"أنلزمكموها".** It is a sentence made up of verb, subject, object, and pronouns. This characteristic makes the application of searching algorithm very interesting. The average length of Arabic words, for example, is 5.7 letters. It is the longest length compared with French words which is 4.84 letters and the 4.5 letters of the English. The number of ASCII codes for the Arabic language is 36 based on Nafitha software developed in Bahrain.

Three experiments have been conducted to calculate the distribution of Arabic letters. The first experiment was perfumed on texts collected from a book. The second experiment was performed on article appeared in an Arabic newspaper based in London known as "Al-Sharaq AlAwsat". The third experiment performed on the whole holly Quran texts. The average has been calculated to draw the distribution as in figure 1.

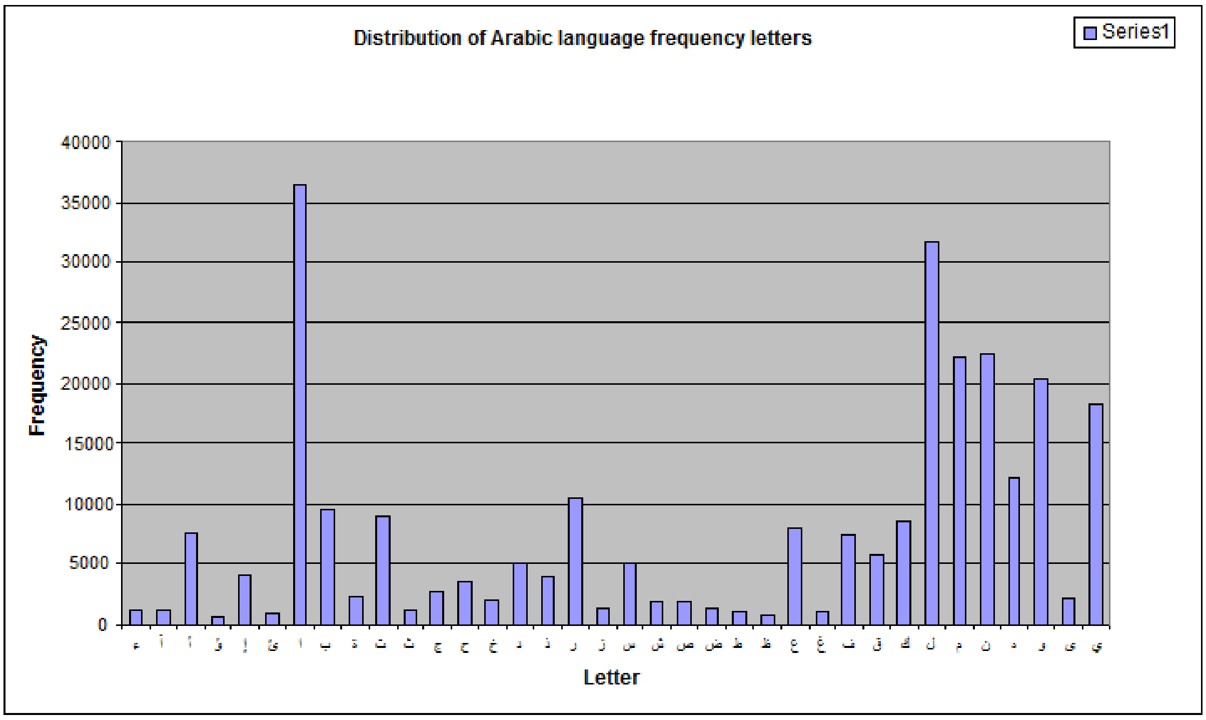

Figure 1: Distribution of Arabic letter frequencies

The frequency can be grouped into four categories starting from the lowest frequency to highest as shown in table 1.

Table 1: Arabic letter frequency groups

Group I	ؤ	ظ	غ	آ	ئ	ء	ث	ز	ط	ض	ش	ص	خ	ى	ج	ة
Group II	إ	ذ	ح	س	د	ق	أ	ف	ك	ع	ب	ت	ر	ه		
Group III	ي	و	ن	م												
Group IV	ل	ا														

The remainder of this paper is organized as follows: the next section presents background information on related work in the area of string matching techniques. It outlines the fastest two algorithms, BMH and BMR. A number of experiments are demonstrated in the empirical investigation section. Finally, the last section draws conclusions and suggests future works.

2. Related Work

The study of detecting the occurrences of a particular sub-string in another string has received much attention from the computer scientists from both theoretical and practical points of views. One of the basic strings searching technique was the straightforward algorithm. In this algorithm, the searching mechanism starts by aligning pattern P against the leftmost portion of the text string T. The algorithm starts comparing the corresponding characters of the pattern and the text from left to right one by one. If at some point during comparisons, a mismatch occurs the pattern P is realigned with T by shifting P one position to the right of T and the comparison processes re-start again.

During the last two decades, several string-searching algorithms have been developed. The earliest two practical algorithms on string searching were the algorithm developed by Knuth, Morris and Pratt (KMP) (1977) and the algorithm developed by Boyer and Moore (BM) (1977). The complexity of both algorithms in the worst case is bounded by Θ (n+km), where k is the total number of matches. there are a number of variations of the BM algorithm available. For example, Boyer-moore-horspool (BMH) (1980) and Raita (1992) are two variants. The performance of these variations and the KMP algorithm are ducumented in smit (1982) and Hume and Sunday (1991). An improved version of BM is well documented in Takaoka (1996).

S. O. Fageeri and R. Ahmad (2014) emphasized the need for improving structure of searching algorithms. They developed a binary-based approach for frequency mining of a database log file. The approach includes the use of algorithms along with its supportive data structures. Construction of the approach began with evaluation of some of the existing methods and identifying their drawbacks. They claim the new algorithms were developed and tested. Initial experimentation of the approach reveals a significant improvement in terms of the execution time of the log file's frequency mining calculation. They did not outlined how fast the execution time compared with other frequently used searching algorithm.

Although the BM algorithm has been analysed extensively, as mentioned above, it is still considered to be the foundation of most of the variations. In the preprocessing phase of the BM algorithm, P is scanned to form two tables. The first table defines a match heuristic where the pattern, P, is matched from right to left.. This table, D, tells how long the pattern can be moved to the left. It is a function of the text character j at which the mismatch occurred.

$D[j] = \min \{k \mid k = m \text{ or } (0 \le k \le m \text{ and } p_{m-k} = j)\}$

The second table expresses the rightmost occurrences of the text symbol x in the pattern. In other word, this table DD, is a function of the position in the pattern at which the mismatch occurred (Crochemore 1997; Lecroq 1995; Berry & Ravindran 1999):

$DD[j] = \min \{k + m - j \mid k >= 1 \text{ and } ((k \ge I \text{ or } p_{i-k} = p_i); j < I \le m)$

$\text{And } (k \ge j \text{ or } p_{j-k} \ne p_j)\}$

The relevant works in the field of string matching (searching) fall into two camps, ones measure the performance by computing the CPU times and the others are based on character comparisons. Our study is based on the character comparisons camp. For the purpose of this study we will review two matching algorithms.

2.1.The Boyer-Moore-Horspool Algorithm

The Boyer-Moore-Horspool algorithm is a variant of the BM algorithm. It uses only one table rather than two tables as in BM. The comparison between the pattern and the text string is made from right to left. The algorithm is expressed as a C++ class abstraction as follows (Horspool 1980):

```
   const AlphabetSize = 256;
           int delta[AlphabetSize];
           int m;
           char* Pat;
 public:
           Search ( char*);
           int find (char*);
 };

 Search :: Search ( char* P)
 {
          assert ( P);
          Pat = P;
          m = strlen(Pat);
          int k = 0;

          for ( k = 0; k < AlphabetSize; k++)
                   delta [k] = m;

          for ( k = 0; k < m-1; k++)
                   delta [Pat[k]] = m -k -1;
   cout<<"\ninside Search"<< delta[Pat[100]];

 }

 int Search :: find ( char* Master)
 {
          assert( Master);
          int n = strlen( Master);
          if ( m > n) return -1;

          int k = m-1;
          while ( k < n) {
               int j = m-1;
```

```
                int i = k;
                while ( j >= 0 && Master[i] == Pat[j]) { j--; i--;}
                if ( j == -1) return i+1;
                k += delta[Master[k]];
            }
            return -1;
    }
```

2.2.Boyer-Moore-Raita Algorithm

Raita (1992) noticed that the BMH algorithm implementation performed well when searching for a random pattern in random texts. In general, text is not random; hence there usually exists strong dependencies between successive symbols (Barnbrook 1996). This variation (BMR) implements a strategy of checking pairs at the end, beginning, and middle of alignment of pattern text. The aim of this variation is to maximize the length of the shifts. The implementation is outlined as follows (Raita 1992):

```
void Raita(char Master[], char pat[], int StringLength, int PatLength)
{
 MidPoint = PatLength/2;
 MidChar = pat[MidPoint + 1];
 MMinusMid = PatLength - MidPoint - 1;
  int k = 0;
   found   = false;
   for( k = 0; k < AlphabetSize; k++) Delta[k]=PatLength;
   for (k = 0 ;k < PatLength-1; k++)  Delta[pat[k]]=PatLength - k-1;
   i = PatLength;
   mminusone = PatLength - 1;
   last = pat[PatLength];
   first = pat[0];
   pat[0]=Symbol_not_in_text; //Replace the first pattern symbol by a sentinel
   while ( (i < StringLength) && !found)
   {
           if (Master[i] == last)
             if (Master[i-mminusone] == first)
               if (Master[i-MMinusMid] == MidChar)
              {
                 k= i -1;
                 j = mminusone;
                while ( Master[k] == pat[j]) {k--; j--; }
                if (j== 1){
                cout<<"\n match at position : "<< k;
                found= true;
               }
     }
    i += Delta[Master[i]];
 }
```

In his paper, Raita [3] described a number of modifications on the BMH algorithm. The improvement was attributed to the existence of character dependencies of the searched text. Smith, in his paper [11] which referred to run time, found that the improvements were due to the addition of guards before the main loop and not to properties of the text being searched. Our experiments on text from Arabic language show that the improved running time on BMR was mainly attributed to the existence of character dependencies and partly to the addition of guards. We present in the next section a number of case studies supporting our findings.

Many of today applications depend heavily on searching and matching techniques. The effectiveness of these applications is influenced by the efficiency of such searching and matching algorithms. For example, text editors and digital libraries or search engines need searching algorithms in order to find patterns in a text. It has also shown in multimedia and computational biology that a much more generalized theoretical basis of pattern matching could be of tremendous benefit. In computational biology, for example, one may be interested in finding a close mutation, in communications one may want to adjust for transmission noise, and in texts it may be desirable to allow common typing errors. In multimedia one may want to adjust for loss compressions, occlusions, scaling, affine transformations or dimension loss. The largest overlap heuristic for finding the shortest common superstring has been used in DNA sequencing.. The time performance of exact string pattern matching, therefore, can be greatly improved when an efficient algorithm is used. Considering the growing amount of text handled in the electronic patient records, it is worth and essential searching for an efficient algorithm to deal with such huge items of information (Singla & Garg 2012).

3. Empirical Investigations

Many variations of the BM algorithm have been investigated in previous works. Boyer and Moore showed that their algorithm makes fewer than i + m comparisons before finding the pattern at location i, That is; the complexity is O(n+km) where k is the total number of matches. The complexity of BMH and BMR is O (cn); c ∊ (0,1). BM variation was implemented using English text. Our experiments use Arabic text, which, unlike English text, has various properties among characters. The two algorithms, BMH and BMR, were implemented in their most efficient forms as suggested by their authors. The most important variables in our experiments are pattern length, finding zero or more occurrences, and whether the text and pattern are random or linguistic.

Case 1

This experiment computes the average number of comparisons for different pattern lengths. Each experiment is repeated n times. The pattern positions and length are chosen arbitrarily from the text.

Organization: The average number of comparisons is computed for each pattern length as follows:

$$Average = \bar{x} = \frac{\sum_{OverallPositions} x_i \quad \times x_i\, positions}{\sum_{OverallPositions} x_i\, positions}$$

Where x_i represents the number of references to the text string before finding the pattern at position i, and $x_i$position represents the position of the first occurrence of the pattern in the text string.

The percentage of saving is computed as follows:

$$Saving = [(\bar{x}BMH \quad - \bar{x}BMR \quad)/\bar{x}BMH \quad]\times 100$$

Algorithm:

Input: Pattern and *Text,* arrays of pattern and text characters; m>0 and n>0, the number of characters in Pattern and Text, respectively.

Processing: calls BMH and BMR algorithms.

Output: The number of references to the text before finding the pattern.

In this first experiment, we investigated a string of length 6930 characters from real Arabic text with different pattern lengths of 5, 6, 9, 15, 25, 27, and 30 characters. The seven-pattern lengths are chosen arbitrarily to make experiments unbiased. Each experiment is repeated four times on different pattern positions. Table 1 shows average number of comparisons required to find occurrences of a pattern.

Table I. Number of Comparisons for BMH and BMR on Arabic texts

Pattern Length	$\bar{x}_{BMH}$	$\bar{x}_{BMR}$	%Saving
5	1316.2	1199.2	8.89
6	972.2	891.4	8.31
9	778.8	746.1	4.20
15	660.8	562.5	14.88
25	535.2	421.1	21.32
27	401.7	376	6.40
30	398.5	386.2	3.09

The results in table I show that the BMR algorithm has better performance than the BMH algorithm. It saves a remarkable amount of computation. The saving range is from 3% to 21%. The variation of the amount of saving indicates that it is caused by the variation of text properties. The saving is displayed in figure (2).

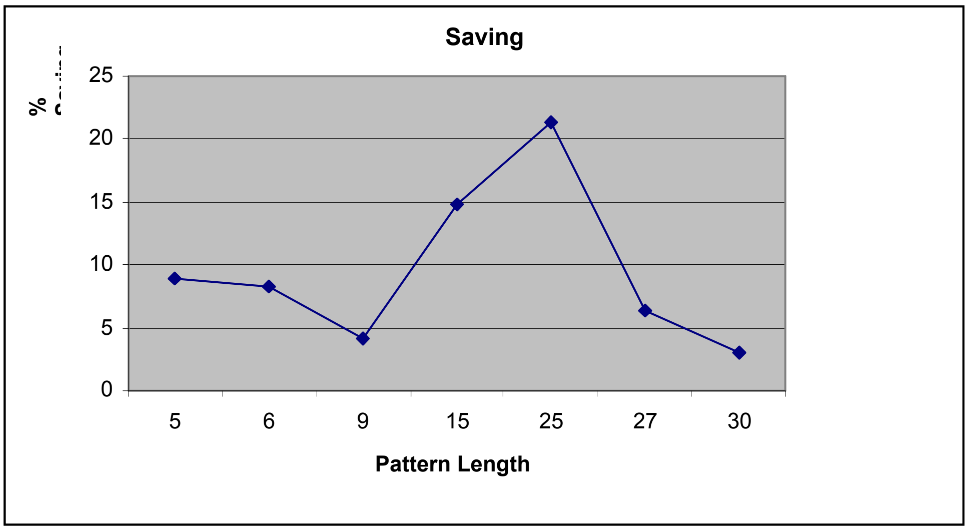

Figure (2): % Saving Costs

In order to do a profound analysis of both algorithms, we conducted n experiments on each algorithm and the results were treated as one sample of n pairs. We computed, for each pair, the difference in performance. We use **confidence intervals (CI)** to guide us during the process of taking a decision. A confidence interval is, in general easier to understand and explain to the decision-makers. Confidence intervals are constructed and along with other measurements are displayed in table 2.

Table 2: Descriptive Statistics for Performance Differences

Measures	Values
Mean	68.7
Standard Error	16.7208
Median	80.8
Standard Deviation	44.23909
Sample Variance	1957.097
Kurtosis	-2.26916
Skewness	-0.19298
Range	104.7
Minimum	12.3
Maximum	117
Sum	480.9
Count	7
Confidence Level(90.0%)	32.49154

From table 2, sample mean = 68.7, sample variance = 1957.097, and sample standard deviation = 44.23909. The 100(1- α)% confidence interval (CI) is given by

$$(\bar{x} - t_{[1-\alpha/2;\, n-1]}\ s/\sqrt{n}\ ,\ \ \bar{x} + t_{[1-\alpha/2;\, n-1]}\ s/\sqrt{n})$$

The $t_{[1-\alpha/2;\, n-1]}$ is the (1- α/2)-quantile of a t-variate with n-1 degrees of freedom. In this case, from table 2, the CI is 32.49154 at the 90% confidence level, i.e.

$$68.7 \pm 32.49154 = (36.20846, 101.19154)$$

The confidence interval does not include zero. The only significant differences between BMH and BMR are the order of character comparisons within the pattern, and if that makes a difference, it must be because it is utilizing some statistical properties of the underlying texts.

Case 2.

This experiment aims to demonstrate the number of comparisons for different pattern lengths ranging from n_1 characters to n_2 characters; $n_1 < n_2$. The pattern has been chosen to be a proper phrase known to be not in text.

Organization: The number of comparisons is computed for each search pattern of length x_k where x_k is defined as follows:

$$x_k = n_l + k \qquad \text{for } k = 0, 1, \ldots, n_2\text{-}n_l \text{ and } x_k \in [n1, n2]$$

The percentage of saving is computed as follows:

Saving = [($x_{BMH} - x_{BMR}$)/ x_{BMH}] × 100

Algorithm

Input: Pattern and Text, arrays of pattern and text characters; m > 0 and n > 0, the number of characters in Pattern and Text, respectively.

Processing: calls BMH and BMR algorithms.

Output: The number of references to the text before finding the pattern.

The patterns length chosen to be a proper Arabic sentence or phrase **"نحن قسمنا بينهم معيشتهم في الح"** that is known to be not in the master text. The search patterns of length m, for each m from 3 to 30. The results are outlined in table 3.

Table 3. Number of Comparisons for BMH and BMR on Arabic texts with patterns not in the text

PatLen	3	4	5	6	7	8	9	10	11	12	13	14	15	16
BMH	2503	2014	1693	1745	1388	1262	1192	1058	1012	1168	862	798	755	706
BMR	2336	1841	1512	1385	1330	1161	1092	1005	916	863	801	762	716	703
Saving	7	9	11	25	4	8	9	5	10	35	7	4	5	0
PatLen	17	18	19	20	21	22	23	24	25	26	27	28	29	30
BMH	636	613	612	720	604	588	553	556	647	498	519	533	611	647
BMR	624	592	554	563	526	548	548	521	503	477	449	513	490	440
Saving	1	3	10	27	14	7	0	6	28	4	15	3	24	47

Some measures of central tendency for the difference between performance of the BMH and BMR algorithms are outlined in table 4.

Table 4: Statistical Measures

Measures	Values
Mean	97.21429
Standard Error	16.75464
Median	65.5
Mode	58
Standard Deviation	88.65721
Sample Variance	7860.101
Kurtosis	2.080643
Skewness	1.454561
Range	357
Minimum	3
Maximum	360
Sum	2722
Count	28
Confidence Level(95.0%)	34.37765

From table 4, the sample size = n= 28, the mode = 58, the mean = 97.21429, and std. Deviation. = 88.65721. The $t_{[1-\alpha/2;\, n-1]}$ is the (1- α/2)-quantile of a t-variate with 27 degrees of freedom. In this case, from table 4, the CI is 34.37765 at the 95% confidence level, i.e.

97.21429 ± 34.37765 = (62.8366, 131.592)

The confidence interval does not include zero. Therefore, there are real differences between the two algorithms' structures. The performance of BMR algorithm cost saving is plotted in figure (3).

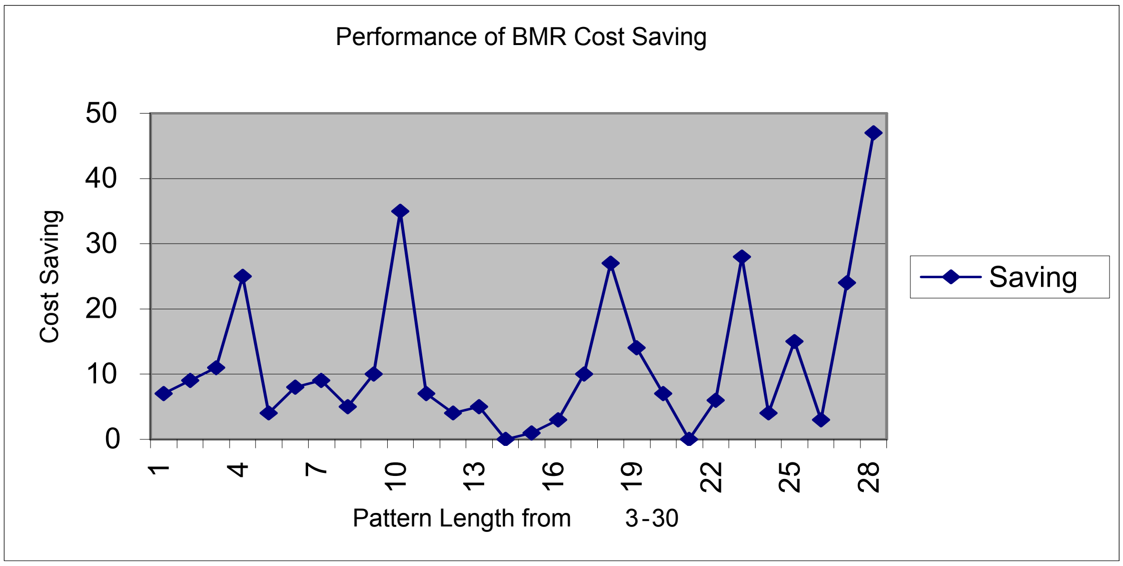

Figure (3): Percentage Saving

The facts displayed by figure (3) lead to the conclusions that improvements are mainly due to character dependencies.

Case 3.

This experiment aims to demonstrate the behavior of the BMH and BMR algorithms on Arabic text with fixed pattern length but with different combinations of letters. The aim is to show whether the properties of the pattern have any effects on the performance of the algorithms.

Organization*:* The number of comparisons is computed for each pattern and for each algorithm. The percentage saving is computed as in case 2.

Algorithm:

Input: Pattern and *Text,* arrays of pattern and text characters; m > 0 and n > 0, the number of characters in Pattern and Text, respectively.

Processing: calls the BMH and BMR algorithms.

Output: The number of references to the text before finding a pattern.

Interested findings were reported in table 5.

Table 5. Number of Comparisons for the BMH and BMR algorithms on Arabic text with equal pattern length and different character types

Pattern	تات	تاي	إلى	بكر	تتت	ممم	///	غنم	قمن	قسم
BMH	2554	2611	2615	2447	2403	2594	2710	2512	2672	2505
BMR	2493	2493	2578	2376	2346	2422	2470	2315	2433	2336
%Saving	**2.4**	**4.5**	**1.4**	**2.9**	**2.4**	**6.6**	**8.9**	**6.4**	**8.9**	**6.7**

Pattern	ملر	رمل	نمر	سشم	قمر	سمس	شمس	نسم	جسم	طلب
BMH	2627	2654	2555	2479	2509	2458	2455	2546	2501	2563
BMR	2547	2438	2482	2323	2433	2429	2424	2380	2332	2488
%Saving	**3**	**8.1**	**2.9**	**6.3**	**3.0**	**1.2**	**1.3**	**6.5**	**6.8**	**2.9**

From table 5, one can see that the saving is varied depending on the type of characters. These investigations may lead to more theoretical and practical studies to develop new variants of the BMH's algorithm. The savings are presented in figure (4).

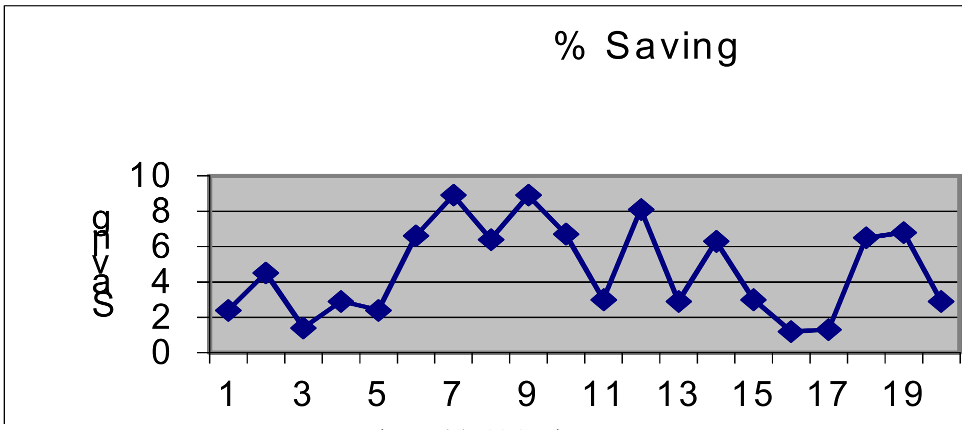

Figure (4): % Saving

In table 6, more investigation is performed using z-test for the difference between the means of the performance of the BMH and BMR algorithms

Table 6: z-test - Two Sample for Means

Measures	*BMH*	*BMR*
Mean	2548.5	2426.9
Known Variance	6820.579	5619.779
Observations	20	20
Hypothesized Mean Difference	0	
Z	4.875646	
P(Z<=z) one-tail	5.43E-07	
z Critical one-tail	1.644853	
P(Z<=z) two-tail	1.09E-06	
z Critical two-tail	1.959961	

We set $\alpha = 0.05$, the critical value for this test, with 18 degrees of freedom is 4.875646. Because z_{observed} (4.875646) > t_{critical} (1.959961), reject the null hypothesis as true. The exact P-value, for z18 = 4.875646, is P = 1.09E-06. In other words, there is a significance difference in the performance of the BMH and BMR algorithms.

Case 4.

This experiment aims to demonstrate the behavior of BMH and BMR algorithms. The master text is of length n. We perform k search patterns of length $m_1,\ldots,m_k$. Experiments were replicated j times at different positions for each pattern of length m_i; i =1,…,k. The first (k-1) repetitions assume successful search deep into the text. The last replica assumes no match occurs. The average number of comparisons per character passed is computed as follows:

$$Average(ChPassed) = \frac{\sum_{i=1}^{k}(Cost_i / Penetration_i)}{k}$$

The master text and the patterns were chosen from Arabic text string of length approximately 15000 characters. The patterns used in the investigations are of length 3,4,6,15,20,24,27, and 30 characters. The eight pattern lengths are chosen arbitrarily to make the experiments unbiased. The experiments have been repeated four times. The first three repetitions assume successful search deep into the master text located roughly at equal intervals. The fourth repetition assumes no match occurs. Hence the algorithm searches all characters of the master text. In each case, the number of comparisons computed and the results for the BMH algorithm were summarized in tables 6.

Table 6: Number of comparisons using BMH algorithm

PatLength	First	Second	Third	Not-Found	Avg(Chpass)
3	793	1954	3920	5805	0.36601
4	608	1627	3024	4347	0.285292
6	486	1229	2276	3152	0.215944
9	343	820	1685	2583	0.158149
15	279	577	1316	1519	0.113074
20	239	646	1027	1273	0.101294
24	331	448	940	1110	0.097244
27	238	554	931	1069	0.091185
30	223	412	779	1059	0.079333
Positions	**2,294**	**5,312**	**11,309**	**14,373**	

Average of comparisons per character passed for pattern, for example, of 3 characters length is:

$$Average(ChPassed) = \frac{1}{4}\left(\frac{793}{2294} + \frac{1954}{5312} + \frac{3920}{11309} + \frac{5805}{14373}\right) = 0.36601$$

Tables 7 summarized the number of comparisons for each pattern length and for each repetition for BMR

Table 7: Number of comparisons using BMR algorithm

PatLength	First	Second	Third	NotFoundd	Avg(Chpass)
3	786	1854	3905	4830	0.346494
4	590	1449	2944	4183	0.265768
6	482	1020	2273	2860	0.198785
9	321	797	1608	2399	0.145548
15	266	575	1254	1499	0.110833
20	227	480	975	1231	0.091473
24	258	439	860	829	0.08845
27	229	406	929	987	0.083709
30	216	387	771	866	0.077011
Positions	**2,294**	**5,312**	**11,309**	**14,373**	

The resultant graph of the average number of comparisons per number of character passed against the pattern length from table 6 and 7 is shown in figure 4.

From figure 4, one can see that the performance of BMR (the lower curve) is better than BMH. One can also see that for patterns of length 4 for example, only 0.15 comparisons are made per character passed. For the BMR algorithm, the number of comparisons also drop to less than 0.1 per character passed or nine-tenth of the characters passed over are not examined when the patterns length, m > 4.

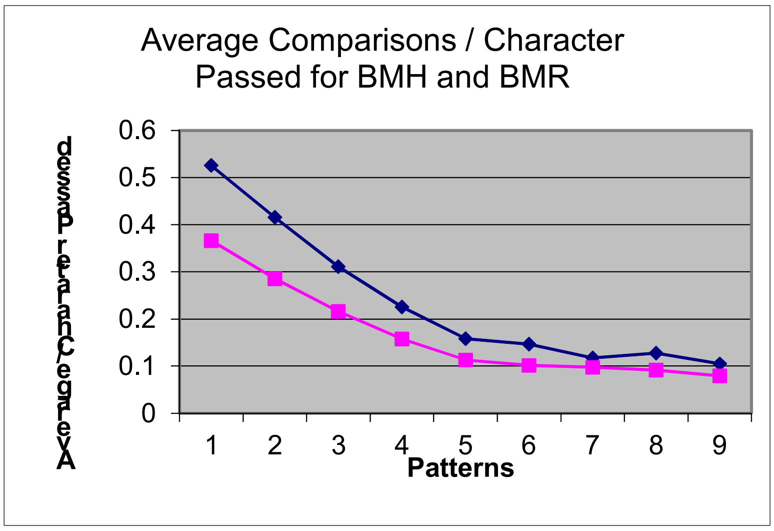

Figure 4: Number of Comparisons / Character Passed

4. Hypothesis testing and Confidence Intervals

For a specified level of confidence (α), a confidence interval specifies plausible values for μ. If those values do not overlap with the value specified by the null hypothesis, it is safe to reject the null hypothesis. Determine if group means differ from each other. State null H_0 and alternate hypotheses H_a:

H_0: Properties of text has no effects. In other words H_0: $\mu_1 = \mu_2$

H_a: There are some effects. i.e. H_a: $\mu_1 \neq \mu_2$

We apply the hypothesis on the average number of comparisons per character passed in table 6 and table 7. The z-test is outlined in table 8:

Table 8: z-Test: Two Sample for Means at 25%

	BMH	*BMR*
Mean	0.237075	0.167503
Known Variance	0.02235	0.010199
Observations	9	9
Hypothesized Mean Difference	0	
Z	1.156887	
P(Z<=z) one-tail	0.123659	
z Critical one-tail	0.67449	
P(Z<=z) two-tail	0.247319	
z Critical two-tail	1.150349	

We set $\alpha = 0.25$, the critical value for this test, with 16 degrees of freedom is 1.150349. Because $z_{observed}$ (1.156887) > $t_{critical}$ (1.150349), reject the null hypothesis as true. The exact P-value, for z16 = 1.156887, is P = 0.247319.

Table 9: t-Test: Two-Sample Assuming Unequal Variances at 5%

	BMH	*BMR*
Mean	0.237075	0.167503
Variance	0.02235	0.010199
Observations	9	9
Hypothesized Mean Difference	0	
Df	14	
t Stat	1.15688	
P(T<=t) one-tail	0.13334	
t Critical one-tail	1.761309	
P(T<=t) two-tail	0.26668	
t Critical two-tail	2.144789	

We believe that the reasons for improvement are due to properties of text being searched. However, a proportion of the improvement is due to the addition of guards before the main loop as experienced by (Smith 1994; Gey & Oard 2002).

Case 5.

In this experiment we have chosen Corpus Data from Arabic newspaper. Total number of characters in the chosen article was 4810. The experiment is repeated for patterns of length 7, 14, 21, 28, and 35. It has been performed for patterns that were known in the master text and for patterns not in the master text. Results were recorded in table 10.

Table 10: Corpus Data from Arabic Newspapers

Pattern Length	Pat In Text			Pat not In Text		
	BMH	BMR	Diff	BMH	BMR	Diff
7	1132	805	327	1363	956	407
14	474	442	32	572	516	56
21	413	373	40	521	432	89
28	484	344	140	591	397	194
35	275	260	15	342	318	24

Overall averages

The overall averages were computed for the case of pattern in the text which shows that the number of comparisons for BMH = 442 and BMR = 355.6. In the mean time, the overall averages were computed for the case of patterns that were not in the text. Result shows that the number of comparisons for BMH = 542.9333 and BMR = 430.8.

Case 6

In this experiment, the BMH and BMR algorithms are used to search for random Arabic text. The text characters were generated randomly and independently from the same character alphabet used in real Arabic text.

Table 11: Search times for BMH and BMR on Random Arabic text

Pattern Length	BMH	BMR	%Saving
3	4068	3946	3
4	5293	5219	1
5	4313	4192	3
6	3676	3564	3
7	3191	3104	3
8	2831	2736	3
9	2544	2462	3
10	2251	2190	3
11	2119	2043	4
12	1976	1911	3
13	1787	1739	3
14	1684	1629	3
15	1590	1552	2
16	1498	1434	4
17	1434	1395	3
18	1346	1310	3
19	1316	1278	3
20	1255	1215	3
21	1194	1146	4
22	1158	1114	4
23	1078	1042	3
24	997	973	2
25	1011	967	4
26	1044	1015	3
27	971	941	3
28	929	908	2
29	923	905	2
30	926	897	3

When we apply statistical measures on table 11, the measures are computed in table 12.

Table 12: statistical measures

Descriptors	Measures
Mean	2.964285714
Standard Error	0.130952381
Median	3
Mode	3
Standard Deviation	0.692934867
Sample Variance	0.48015873
Kurtosis	1.471700231
Skewness	-0.672485365
Range	3
Minimum	1
Maximum	4
Sum	83
Count	28
Confidence Level (95.0%)	0.268691911

The results, in table 12, indicate that the percentage saving is nearly constant with approximately equal values for the mean, mode, and median. These results support our conclusions that improvement is mainly because of the text character properties which are supported by (Gurung et al. 2016).

5. Conclusions

In today's applications finding the appropriate content in minimum time is very important. String searching algorithms perform a vital role for this. Researchers, all over the world, are hard working on software and hardware levels to make pattern searching faster. The aim is to find algorithms that can be able to reduce complexity and also reduce computation time. it has been noted that most applications uses Boyer Moore, BMH or BMR variant algorithms for their effective and efficient functionality and other applications uses the basics of these algorithms for their functionalities as the BMR algorithm has less time complexity and Boyer Moore and BMH algorithms has preprocessing time complexity less. Other algorithms depends upon the type of input and is efficient for certain or particular application.

The work in this paper evaluates the behavior of two variants of matching algorithms BMH and BMR. The paper outlined the techniques used in the comparisons. A number of experiments were conducted and results were recorded in tables. The results show that on average the number of comparisons per character passed in the case of the BMR is less than the number encountered by the BMH variants. The improvement is due to properties of the text structures.

These experiments may lead to more theoretical and practical studies to develop new variants of algorithms. Using Arabic text structures provided more insight into the structure of the algorithm as the Arabic language has a different set of characters and its characters properties have different structures. Improvement on string searching algorithms has many real world applications including the extraction of relevant data from web pages.

References

Knuth, D. E., Morris Jr, T. H. & Pratt, V. B. (1997), "Fast Pattern Matching in Strings", SIAM J. Computing, Vol. 6, No. 2, pp. 323-350, 1977.

Boyer, R. S. & Moore, J. S. (1977), "A fast String Searching Algorithm", Communications of the ACM, Vol. 20, No. 10, pp. 762-772, October 1977.

Horspool, R. N. (1980, "Practical Fast Searching in Strings", Software Practice and Experience, **10,** pp.501-506, 1980.

Raita, T. (1992), "Tuning the Boyer-Moore-Horspool String Searching Algorithm", Software Practice and Experience, Vol. 22, No. 10, pp.879-884, 1992.

Smit, G. (1982), "A Comparison of Three String Matching Algorithms", Software-Practice and Experience, Vol. 12, pp.57-66, 1982.

Hume, A. & Sunday, D. M. (1991), "Fast String Searching", Software-Practice and Experience, 21, pp.1221-1248, 1991.

Takaoka, T. (1996), "A Left-to-Right Preprocessing Computation for the Boyer-Moore String Matching Algorithm", The Computer Journal, Vol. 39, No. 5, 1996.

Crochemore, M. (1997), "Off-line Serial Exact String Searching", in Pattern Matching Algorithms, A. Apostolico and Z. Galil ed., Chapter 1, pp 1-53, Oxford University Press, 1997.

Lecroq, T. (1995), "Experimental Results on String Matching Algorithms", Software Practice and Experience, Vol. 25, No. 7, pp. 727-765, 1995

Berry, T. & Ravindran, S. (1999), " A Fast String Matching Algorithm and Experimental Results", Proceedings of the Prague Stringology Club Workshop'99, J. Holub and Simanek ed., Collaborative Report DC-99-05, Czech Technical Univ., Prague, Czench Republic, pp. 16-26, 1999.

Smith, P. D. (1994), "On Tuning the Boyer-Moore-Horspool String Searching Algorithm", Software-Practice and Experience, Vol. 24 (4), 435-436 (April 1994).

Barnbrook, G. (1996), "Language and computers: A practical Introduction to the Computer Analysis of Language", Edinburgh University Press, 1996.

Gey, F. C. & Oard, D. W. (2002), The TREC-2001 cross-language information retrieval track: Searching Arabic using English, French, or Arabic queries. In *TREC 2001*. Gaithersburg: NIST, 2002.

Xu, J., Fraser, A. &Weischedel, R. (2002), Empirical studies in strategies for Arabic retrieval. In *Sigir 2002*. Tampere, Finland: ACM, 2002.

Young-Suk Lee, Kishore Papineni, Salim Roukos, Ossama Emam & Hany Hassan (2003), Language Model Based Arabic Word Segmentation, for Computational Linguistics, July 2003, pp. 399-406. Proceedings of the 41st Annual Meeting of the Association.

Fageeri, S. O. & Ahmad, R. (2014), An Efficient Log File Analysis Algorithm Using Binary-based Data Structure, Procedia - Social and Behavioral Sciences, Volume 129, 15 May 2014, Pages 518–526.

Gurung, D. , Chakraborty, U. & Sharma, P. (2016), Intelligent Predictive String Search Algorithm, 7th International Conference on Communication, Computing and Virtualization, At Mumbai, 2016.

Singla, N. & Garg, D. (2012). String Matching Algorithms and their Applicability in various Applications International Journal of Soft Computing and Engineering (IJSCE), Volume-I, Issue-6, January