# A Survey on Bit Parallel String Matching Algorithms

**Hiresh Singh Sengar**
**M Tech Scholar, Department of CSE**
**JNCT, Bhopal INDIA**
**E Mail:- hireshtech@gmail.com**

**Prof. B.L. Rai**
**DEAN, ADMINISTARTION**
**JNCT, Bhopal INDIA**
**E Mail:- blrai_08_76@yahoo.com**

**Prof. Deepak Jain**
**HOD, Department of CSE**
**JNCT, Bhopal INDIA**
**E Mail:- deepak.jain25@gmail.com**

**ABSTRACT**
String matching algorithms plays an important role in real life example of computer science engineering. The searching time of pattern is an overhead so requirement of the fast pattern searching is needed. Character based searching is limited upto certain level due to this Bit Parallel String Matching Algorithms comes into existence. The intrinsic parallelism in bit operations like AND/OR inside a computer word is known as bit parallelism. Since 1992, this bit parallelism is directly used in string matching for matching efficiency improvement. Some of the popular bit parallel string matching algorithms Shift OR, Shift OR with Q-Gram, BNDM, TNDM, SBNDM, LBNDM, FBNDM, BNDMq, and Multiple pattern BNDM. This paper discusses the working of various bit parallel string matching algorithms with example. Here we present how bit parallelism is useful for efficiency improvement in various algorithms with their detailed explanation and comparative analysis between these algorithms**.**

**Keyword:- String Matching Algorithm, Bit Parallel string matching, Shift OR, BNDM, TNDM, Shift OR with Q gram, BNDMq, Filtering based Multiple BNDM.**

## INTRODUCTION

Bit parallelism [1] is an intrinsic property of computer in which bit operations are performed parallely within the computer word in the single clock. With the use of bit parallelism in String Matching algorithm speed of matching is improved up to certain level. String matching [2] algorithms are used in most of the real world applications where pattern extraction is required like as Intrusion Detection system [3][4], Plagiarism detection [5], Data Mining [6] and Bioinformatics [7]. Bit parallel algorithms are faster than the other benchmark character based algorithms like as KMP [4][8], BM [9][10], BMH [11][12], BMHS[13], BMHS2[14], BMI[15], Improved BMHS[16], Cmmentz Walter[17][18], Wu Manber[19][20] and Aho-Corasick [21][22] etc. Bit Parallel algorithms [23] are based on the non-deterministic automata but there is no such automata are present. It is simply the efficient simulation of non-deterministic automata. Figure-1 shows how the bit parallel operations are performed inside the computer word. Here computer word size length is 8 bits or 1 byte.

**Computer Word 1:**

| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|

**Computer Word 2:**

| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|

**Resultant Word 3:**

| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|

Figure 1: Operation perform inside the computer

After performing “&” operation at bit level in parallel, the resultant word can be obtained like

First bit parallel algorithm was introduced in 1992 by the Baeza–Yates and Gonnet named as Shift OR algorithm [24]. It was a approximate multiple pattern string matching algorithm. It was faster than the previous algorithms but gives false matches.

After Shift OR in 1998 Navarro and Raffinot were introduced new bit parallel algorithm named as BNDM (Backward Non Deterministic Matching)[25]. This was an exact single pattern string matching algorithm. In BNDM algorithm we use AND or SHIFT operation which performed in parallel. BNDM is faster than character based algorithms but here pattern size must be smaller or equal than the computer word size.

BNDM set the benchmark in the string matching algorithm. After BNDM in 2003 Peltola and Tarhio introduced improved version of BNDM known as TNDM (Two way Non Deterministic Matching)[26]. In TNDM scanning is much similar to the BNDM except mismatch at last position instead of shifting it scan forward.

In 2003 another algorithm was introduced Simplified BNDM also known as SBNDM [26]. It was an improved version of BNDM. Here we do not require finding the longest prefix that's why the average length of shift is reduced. Here the inner most loop becomes simpler.

In 2005 Longtao He, Binxing Fang and Jie Sui proposed a bit parallel algorithm known as Wide Window Algorithm [27]. It use the wide window of size one less than the two time the pattern length to attempt m position in parallel. They combine the bit parallel technique with new wide window concept. It is exact single pattern matching algorithm which is faster in most of the cases.

In 2006 an improved version of Shift OR was introduced by Salmela, Tarhio and Kytojoki known as Shift OR with Q-Gram [28]. Same as Shift OR it is an approximate multiple string matching algorithm. It considers Q character at a time for comparison by doing that the size of automata is reduced and number of comparisons for finding pattern is reduced up to certain level. This algorithm also reduces the false matches.

In 2009 Branislav Durian, Jan Holub, Hannu Peltola and Jorma Tarhio introduced the concept of Q gram in BNDM algorithm known as BNDMq [29]. It reads the q characters at each alignment before testing the state variable. So most of the cases the number of comparisons are less than in comparison to BNDM required. Similarly as BNDMq Branislav Durian, Jan Holub, Hannu Peltola and Jorma Tarhio also introduced SBNDMq [29] in 2009. It is an exact single pattern string matching algorithm. Similar approach of BNDMq was used in SBNDMq which gives the better results in comparison to SBNDM.

In 2010 Changsheng Miao, Guiran Chang and Xingwei combine the concept of Q gram with BNDM and implement exact multiple string matching algorithm known as Filtering based multiple string matching algorithm [30].

Table 1 shows the evolution of bit parallel algorithm with their description.

**Table 1: Evolution of Bit Parallel Algorithm**

| S.No. | Algorithm | Year | Authors |
|---|---|---|---|
| 1 | Shift OR | 1992 | Baeza-Yates and Gonnet |
| 2 | BNDM | 1998 | Navarro and Raffinot |
| 3 | TNDM | 2003 | Peltola and Tarhio |
| 4 | SBNDM | 2003 | Peltola and Tarhio |
| 5 | Wide Window | 2005 | Longtao He, Binxing Fang and Jie Sui |
| 6 | Shift OR with Q Gram | 2006 | Salmela, Tarhio and Kytojoki |
| 7 | BNDMq | 2009 | Branislav Durian, Jan Holub, Hannu Peltola and Jorma Tarhio |
| 8 | Multiple BNDM with q gram | 2010 | Changsheng Miao, Guiran Chang and Xingwei |

Sting Matching using Bit Parallelism can be classified into two broad categories that are single pattern matching and multiple pattern matching. In single pattern matching there is a single pattern which is searched in to the text and reports occurrences. In multiple pattern matching we have number of pattern whose occurrences we have to report. As compared to single pattern matching multiple patterns matching algorithm has lots of practical application in real life.

## II SINGLE PATTERN BIT PARALLEL ALGORITHM

Single pattern matching means finding the pattern in the text and reports the occurrences of the pattern. We have various single pattern bit parallel string matching algorithm which are discussed one by one below with the help of the example.

### BNDM ALGORITHM

BNDM stands for Backward Non Deterministic Matching [25]. It is exact single pattern string matching algorithm. In BNDM order of searching is from right to left. It uses the concept of bit parallelism from shift OR algorithm [24] and suffix automata from BDM algorithm [25]. This algorithm is a bit parallel simulation of BDM algorithm. BDM skips character using suffix automata which is deterministic in pre-processing. To construct Deterministic automata is complex task. BNDM simulates the non-deterministic version using bit parallelism. BNDM algorithm consist in two phase Pre-processing phase and Searching phase

**Pre-processing Stage:** In pre-processing phase we find Bit Vector of each Character of the Pattern calculated by putting 1 for occurrence and 0 for non-occurrence and take bit vector D with initial value with all one.

**Searching Stage:** In searching phase pattern is searched with the help of two logical operator that are AND and SHIFT. Pattern is searched when MSB of D is 1 and value of j is 0.

Let us understand the working of whole algorithm with the use of an example. Example of BNDM is as follow. Let Text T = 'SPRINGCARE' and Pattern P= CARE Bit vector of character calculated in preprocessing stage B[C] =1000, B[A] =0100, B[R] =0010, B[E] =0001 and B[OTHER] =0000 and in searching stage Initially we take some variable and set to the pattern length i.e. J=4, last=4, pos. = 0 and bit vector D =1111. Than we perform the BNDM algorithm whose various step are shown in the Table 2 with their explanation of each steps.

**Table 2: Shows the various steps perform by BNDM**

| SCANNING PHASE | | | | | | | |
|---|---|---|---|---|---|---|---|
| STEP | TEXT | D | B[k] | D' | J | LAST | MATCHING |
| 1 | SPRINGCARE | 1111 | - | - | 4 | 4 | |
| 2 | SPRINGCARE | 1111 | 0000 | 0000 | 3 | 4 | |
| 3 | SPRINGCARE | 1111 | - | - | 4 | 4 | |
| 4 | SPRINGCARE | 1111 | 0100 | 0100 | 3 | 4 | |
| 5 | SPRINGCARE | 1000 | 1000 | 1000 | 2 | 2 | |
| 6 | SPRINGCARE | 0000 | 0000 | 0000 | 1 | 2 | |
| 7 | SPRINGCARE | 1111 | - | - | 4 | 4 | |
| 8 | SPRINGCARE | 1111 | 0001 | 0001 | 3 | 4 | |
| 9 | SPRINGCARE | 0010 | 0010 | 0010 | 2 | 4 | |
| 10 | SPRINGCARE | 0100 | 0100 | 0100 | 1 | 4 | |
| 11 | SPRINGCARE | 1000 | 1000 | 1000 | 0 | 4 | CARE |

BNDM algorithm become very fast string matching algorithm except very sort (0-6) or very long (90-150) pattern. It is faster than the previous algorithm Shift OR, BDM and Occupies very less space perform various operation in parallel. It is very Simple and flexible algorithm. But we have all pattern assume to less than or equal to the word size of computer.

**TNDM ALGORITHM**

TNDM stands for Two ways Non Deterministic Matching which is introduced by Peltola and Tarhio in 2003[26]. It is Exact Single pattern string matching algorithm. Here order of scanning is from left to right. It is almost same as the BNDM algorithm there is slight changes in the case of mismatch occur at first position instead of shifting TNDM look forward to find suffix of reverse pattern. The number of examined characters is less than BNDM therefore matching is faster. The simulation of TNDM Algorithm can be understood by the help of the Figure 2.

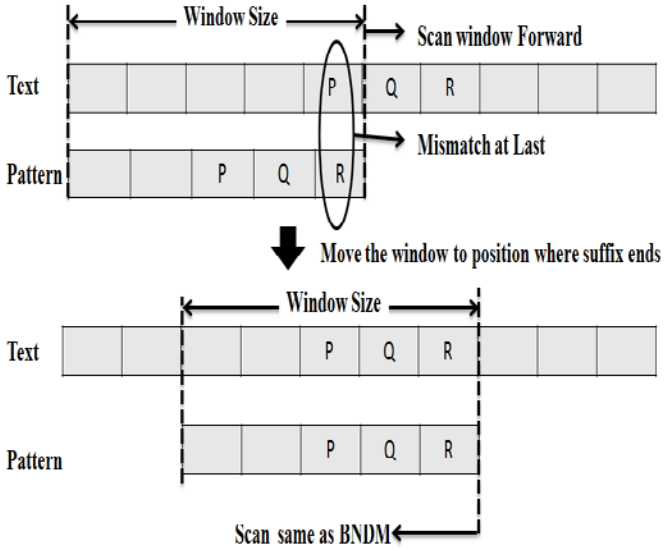


**Figure 2: Working explanation of TNDM algorithm**

TNDM uses bit parallelism through which it become faster algorithm. It uses forward scan so the number of examined character is less than the BNDM in general. Remaining algorithm is work same as BNDM algorithm where all pattern assume to less than or equal to the word size of computer.

**SBNDM ALGORITHM**

SBNDM [26] stands for Simple Backward Non Deterministic Matching. SBNDM is much similar to the BNDM algorithm there is a slight change in term of shifting. Due to this it is quit faster than the BNDM algorithm. Here we do not require finding the longest prefix. Let T is the text of length n and P is the pattern of length m to be searched. At each alignment window of P in T, Scan T from right to left until the suffix of the window is not a factor of P or an occurrence of P is found. Shifting of SBNDM is according to this cases i.e. shift window by m if no factor is found, shift by 1 if P found otherwise next alignment is start at last factor. Table 3 describes the various steps of the SBNDM algorithm.

**Table 3: Working Explanation of SBNDM**

**Pattern= 'DESIGN', Text = 'SFZIGNBACDESIGN'**

| ALIGNMENT | S | F | Z | I | G | N | B | A | C | D | E | S | I | G | N |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | N | | | | | | | | | |
| | | | | | G | N | | | | | | | | | |
| | | | | I | G | N | | | | | | | | | |
| NOT A FACTOR | | | Z | I | G | N | | | | | | | | | |
| NEXT ALIGNMENT | S | F | Z | I | G | N | B | A | C | D | E | S | I | G | N |
| NOT A FACTOR | | | | | | | | | C | | | | | | |
| NEXT ALIGNMENT | S | F | Z | I | G | N | B | A | C | D | E | S | I | G | N |
| PATTERN FOUND | | | | | | | | | | D | E | S | I | G | N |

By using the SBNDM concept the average length of shift is reduced by doing so the innermost loop of algorithm become simpler. It is faster than the BNDM algorithm.

**FORWARD SBNDM ALGORITHM**

Forward SBNDM [26] is also known as FSB introduced by Faro and Lecroq. It is enhanced version of the BNDM algorithm which use the Non Deterministic Automaton augmented of new initial state in order to take into account the forward character of the current window of the text. FSB reads a 2-gram x1x2 before a factor test. Only x1 is matched with the end of P in FSB and x2 is a look ahead character.

Let UV be a q-gram, where |V| = f. After reading UV there are 3 alternatives:

If U is a suffix of P, reading continues leftwards.

Else if UV is a factor of P, reading continues leftwards.

Else the state vector is zero and P is shifted m-q+f+1 positions

**BNDM WITH Q-GRAM**

BNDM with Q-gram [26] is an improved variation of the BNDM algorithm which reads the q gram at each alignment before testing the state variable. In this algorithm loop has been made as sort as possible in order to quickly advance m-q+1 position. Here q can be varies according to our requirement. The whole algorithm can be easily understood with the help of the example given below.

Let Text T = 'SPRINGCARE' and Pattern P= 'CARE' Assume we have 2-gram

**Pre-processing:** In pre-processing phase we find Bit Vector of each Character of the Pattern calculated by putting 1 for occurrence and 0 for non-occurrence.

B[C] =1000, B[A] =0100, B[R] =0010, B[E] =0001 and B[OTHER] =0000

**Searching Phase:** Initially we take variable i and set to m-q+1 where m is pattern length

Than we perform the BNDMq algorithm whose various step shown in the table 4.

**Table 4: working of BNDMq algorithm**

| SCANNING PHASE | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| STEP | TEXT | I | D | B[k] | D' | J | FIRST | MATCHING |
| 1 | SPRINGCARE | 3 | 0000 | - | - | - | - | |
| 2 | SPRINGCARE | 6 | 0000 | - | - | - | - | |
| 3 | SPRINGCARE | 9 | 0010 | 0010 | 0100 | 8 | 6 | |
| 4 | SPRINGCARE | 9 | 0100 | 0100 | 1000 | 7 | 6 | |
| 5 | SPRINGCARE | 9 | 1000 | 1000 | 0000 | 6 | 6 | CARE |

It is become very fast string matching algorithm except very sort (0-6) or very long (90-150) pattern. It is faster than the previous algorithm BNDM and Occupies very less space and simple to implement.

### III ULTIPLE PATTERN BIT PARALLEL ALGORITHM

Multiple pattern string matching means we have number of pattern and text so we have to find out the occurrence of these pattern in the text. We have various bit parallel multiple string matching algorithm. In this section we discuss these multiple algorithm one by one with the help of example.

### SHIFT OR ALGORITHM

It is a first algorithm which uses the concept of bit parallelism introduced by Baeza Yates and Gonnet in 1992[24]. It is a approximate multiple pattern string matching algorithm which means it search number of pattern at a time but there is a possibility of error. In Shift OR algorithm order of searching is from left to right. It is design for large pattern have equal length and equal or less than the word size.

Many multiple pattern algorithm build a trie of the pattern in pre-processing phase so as the pattern size increases size of tree also increase which is not practical to maintain. Shift OR algorithm is a simulation of nondeterministic automata where we do not need to build any trie. They don't need to buffer the input. It is real time algorithm suitable to be implemented in hardware. Shift OR consist into two phase:

1. Pre-Processing Phase 2. Searching Phase

Let's take an example to understand Shift OR algorithm. Suppose CARE and TINB be the patterns of length 4 and SPRINGCARE be the text.

**Pre-processing phase:** In pre-processing phase we find out the bit vector of the every character of alphabet. In this '$i_{th}$' bit is zero if and only if character appears at position 'i' otherwise place 1 and write it into reverse order.

B[C] = 1110, B[A] = 1101, B[R] = 0100, B[E] = 011, B[T] = 1110, B[I] = 1101, B[N] = 1011, B[B] = 0111 and B[OTHER] = 1111.

**Searching Phase:** The automation has a transition from state 'i' to 'i+1' on character c if and only if '$i_{th}$' bit in B[c] is 0. In state vector D where '$i_{th}$' bit is 0 if and only if state 'i' in the automation is active. If 0 is occurs at MSB means we find the pattern at position. Occurrence position = pos - pattern length + 1. All the steps involved are shown in the table 5.

**Table 5: Working of Shift OR algorithm**

| SCANNING PHASE | | | | | | |
|---|---|---|---|---|---|---|
| STEP | TEXT | D | B[k] | D' | POSITION | MATCHING |
| 1 | SPRINGCARE | 1111 | - | - | - | |
| 2 | SPRINGCARE | 1110 | 1111 | 1111 | 1 | |
| 3 | SPRINGCARE | 1111 | 1110 | 1110 | 2 | |
| 4 | SPRINGCARE | 1101 | 1111 | 1111 | 3 | |
| 5 | SPRINGCARE | 1111 | 1101 | 1111 | 4 | |
| 6 | SPRINGCARE | 1111 | 1011 | 1111 | 5 | |
| 7 | SPRINGCARE | 1111 | 1111 | 1111 | 6 | |
| 8 | SPRINGCARE | 1111 | 1110 | 1110 | 7 | |
| 9 | SPRINGCARE | 1101 | 1101 | 1101 | 8 | |
| 10 | SPRINGCARE | 1011 | 1011 | 1011 | 9 | |
| 11 | SPRINGCARE | 0111 | 0111 | 0111 | 10 | CARE |

In Shift OR algorithm pre-processing and search are very simple and only bitwise logical operations Shift and AND are used and no Buffering is required. It is an real time algorithm. Time delay to process one text character is bounded by a constant depend only on pattern length. It is approximate string matching algorithm so possibility of the error (false match) and Here all pattern length must be equal.

### SHIFT OR WITH Q-GRAM

Shift OR with Q-gram [28] is an enhanced version of the Shift OR algorithm. In Shift OR with Q-garm algorithm we take q character at a time for comparison by doing so the size of automata is reduced and the number of comparison for finding pattern is reduced up to certain level. The Q-gram can be of two types: Consecutive Q-gram or Overlapped Q-gram. In Consecutive Q-gram WE read pattern in a sequence of q character at a time while in Overlapped Q-gram we take q character from each character of the patterns. Here the pattern length of each pattern must be same. Shift OR with Q-gram carried out in three phase First phase: Initialization Phase where initialization of the variable is carried out. Second phase: Pre-processing phase where bit vector of the various q gram are taking place. Third phase: Searching phase here searching of the pattern in the text is carried out.

Let's take an example of Shift OR Consecutive 2-Gram where Text is STRINGCARE and Patterns 'CARE' and 'TINB'

hence the consecutive 2-gram of first pattern is "CA" and "RE" and second pattern is "TI" and "NB" by doing so the number of bit in bit vector is reduced to the half of the pattern length. Here 2-gram of the pattern is treated as single character. The ith bit of the bit vector is set to zero if there is an occurrence of 2-gram in the ith position of the pattern otherwise set to one. So bit vector of our pattern is as follow.

B [CA] = 10, B [RE] = 01, B [TI] = 10, B [NB] = 01 and B [OTHER] = 11.

Initialise D = 11 and Update D when a character c is read from the text by

D= (D<<1) | B[c]

Various steps involved are shown in table 6.

**TABLE 6: WORKING OF SHIFT OR WITH QGRAM**

| SCANNING PHASE | | | | | |
|---|---|---|---|---|---|
| STEP | TEXT | D | B[k] | D' | MATCHING |
| 1 | SPRINGCARE | 11 | - | - | |
| 2 | SPRINGCARE | 10 | 11 | 11 | |
| 3 | SPRINGCARE | 10 | 11 | 11 | |
| 4 | SPRINGCARE | 10 | 11 | 11 | |
| 5 | SPRINGCARE | 10 | 11 | 11 | |
| 6 | SPRINGCARE | 10 | 11 | 11 | |
| 7 | SPRINGCARE | 10 | 11 | 11 | |
| 8 | SPRINGCARE | 10 | 10 | 10 | |
| 9 | SPRINGCARE | 01 | 01 | 01 | CARE |

**MULTIPLE PATTERN BNDM BY COMBINING Q-GRAM**

BNDM algorithm is a single pattern string matching algorithm which is very fast one because it uses the concept of bit parallelism. Changsheng Miao, Guiran Chang and Xingwei Wang convert the BNDM algorithm in multiple pattern BNDM algorithms [18]. They develop Filtering Based Multiple String Matching Algorithm by Combining q-Grams and BNDM.

**IV COMPARISION AND ANALYSIS**

Above we describe the general bit parallel string matching algorithms in detail. Each of those algorithms works on the concept of the bit wise operator but these algorithms differ in several case. Here the table 7 gives the detail comparison of the various bit parallel string matching algorithm based on various parameter.

**Table 7: Comparison of Various Wu Manber String Matching Algorithms**

| Parameter / Algorithm | Operator Used | Scanning | Pattern | Number of Character Read | Maximum Shift | Size of Patterns |
|---|---|---|---|---|---|---|
| SHIFT OR ALGORITHM | OR | Left to Right | Multiple Pattern | Single | 1 | EQUAL |
| SHIFT OR WITH Q-GRAM ALGORITHM | OR | Left to Right | Multiple Pattern | Q-Character | 1 | EQUAL |
| BNDM ALGORITHM | AND | Right to Left | Single Pattern | Single | M | - |
| TNDM ALGORITHM | AND | Right to Left | Single Pattern | Single | M | - |
| FILTERING BASED MULTIPLE BNDM ALGORITHM | AND | Right to Left | Multiple Pattern | Q-Character | M-Q+1 | EQUAL |

**V CONCLUSIONS**

After development of the processor technology, a whole new series of algorithms has been started for the string matching. Since, string matching plays a big role in applications like DNA matching, plagiarism, data mining; web searching etc. the research is going on all around the world to develop fast and efficient string matching algorithms. The development of Bit Parallel has set a new milestone in this field. Till time many new algorithms, especially those applying bit-parallelism, has been introduced which are much faster than the old ones. When comparing the search speed of two string matching algorithms, several factors affect the result like processor, compiler, and stage of tuning, text, and pattern. Even a small change in the pattern may switch the order of the algorithms. Thus there is no absolute truth which algorithm is better. Because the continuing development of processor and compiler technologies, it is also difficult to anticipate, how present algorithms manage after a few years.

**REFERENCES:-**

[1] Vidya Saikrishna, Akhtar Rasool and Nilay Khare, "Time Efficient String Matching Solution for Single and Multiple Pattern using Bit Parallelism", In procd. Of International Journal of Computer Applications (0975 – 8887) Volume 46–No.6, May 2012.

[2] Christian Charras and Thierry Lecroq," Handbook of Exact String_Matching Algorithms", Published in King's college publication, Feb 2004.

[3] Ali Peiravi, "Application of string matching in Internet

Security and Reliability", Marsland Press Journal of American Science, 6(1), pp. 25-33, 2010.

[4] Pei-fei Wu and Hai-juanShen,"The Research and Amelioration of Pattern-matching Algorithm in Intrusion Detection System", In the proc. of IEEE 14th International Conference on High Performance Computing and Communication & IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS), pp. 1712-1715, 25-27 June 2012.

[5] Ramazan S. Aygün "structural-to-syntactic matching similar documents", Journal Knowledge and Information Systems, ACM Digital Library, Volume 16 Issue 3, pages 303-329, Aug 2008.

[6] Sanchez D., Martin-Bautista M.J., Blanco I. and Torre C.," Text Knowledge Mining: An Alternative to Text Data Mining", In the proc. of IEEE International Conference on Data Mining Workshops, ICDMW '08, pp. 664-672, 15-19Dec. 2008.

[7] Robert M. Horton,Ph.D. "Bioinformatics Algorithm Demonstrations in Microsoft Excel", California State University, Sacramento, 2004.

[8] Knuth D E, Morris Jr J. H and Pratt V. R," Fast pattern matching in strings", In the procd. Of SIAM J.Comput., Vol. 6, 1, pp. 323–350, 1977.

[9] Boyer R S and Moore J S,"A fast string searching algorithm", Communication of ACM 20, Vol. 10, pp. 762–772, 1977.

[10] Zhengda Xiong,"A Composite Boyer-Moore Algorithm for the String Matching Problem", In the proc. of International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT), pp. 492-496, 8-11 Dec 2010.

[11] Horspool R N,"Practical fast searching in strings", In proc. Of Software Practical Exp, Vol. 10, 6, pp. 501–506, 1980.

[12] Timo Raita,"Tuning the Boyer–Moore–Horspool String Searching Algorithm", In the proc. of Software Practice and Experience, Vol. 22(10), pp. 879–884, Oct. 1992.

[13] JingboYuan, Jinsong Yang and Shunli Ding," An Improved Pattern Matching Algorithm Based on BMHS", In the proc. Of 11th International Symposium on Distributed Computing and Applications to Business, Engineering & Science, 2012.

[14] Yuting Han and Guoai Xu, "Improved Algorithm of Pattern Matching based on BMHS", In the proc. of IEEE International Conference on Information Theory and Information Security (ICITIS), pp. 238-241, 17-19 Dec 2010.

[15] Jingbo Yuan, Jisen Zheng and Shunli Ding, "An Improved Pattern Matching Algorithm", In the proc. of Third International Symposium on Intelligent Information Technology and Security Informatics (IITSI), pp. 599-603, 2-4 April 2010.

[16] Linquan Xie, Xiaoming Liu and Guangxue Yue, "Improved Pattern Matching Algorithm of BMHS", In the proc. of International Symposium on Information Science and Engineering (ISISE), pp. 616-619, 24-26 Dec 2010.

[17] Commentz-Walter, "A string matching algorithm fast on the average," In the Proc. of 6th International Colloquium on Automata, Languages, and Programming, pp. 118–132, 1979.

[18] Kouzinopoulos, C.S. and Margaritis, K.G.,"A Performance Evaluation of the Pre-processing Phase of Multiple Keyword Matching Algorithms", In the proc. of 15th Panhellenic Conference on Informatics (PCI), pp. 85-89, 30 Sept 2011- 2 Oct 2011.

[19] Yang Dong hong, XuKe and Cui Yong,"An improved Wu-Manber multiple patterns matching algorithm", In the proc. Of 25th IEEE International Performance, Computing, and Communications Conference, IPCCC, pp. 680, 10-12 April 2006.

[20] Baojun Zhang , XiaoPing Chen , Lingdi Ping , Wu, Zhaohui, "Address Filtering Based Wu-Manber Multiple Patterns Matching Algorithm", In the proc. of 2009 Second International Workshop on Computer Science and Engineering[WCSE], Qingdao, Vol.1, pp. 408 – 412,28-30 Oct. 2009.

[21] Alfred v aho and Margaret j corasick,"efficient string matching: an aid to bibliographic search" communication of acm, vol. 18, June 1975.

[22] Tao Tao and Mukherjee A.,"Multiple-pattern matching in LZW compressed files using Aho-Corasick algorithm", In the proc. of Data Compression Conference, 21-31 March 2005.

[23] Faro S. and Lecroq T,"The exact online string matching problem: A review of the most recent results", ACM Comput. Survey, Article 13, 42 pages, February 2013.

[24] Ricardo A. Baeza-Yates and Gaston H. Gonnet,"A New Approach to Text Searching", In Communications of the ACM, pp. 74-82, Oct 1992.

[25] G. Navarro and M. Raffinot, "Fast and flexible string matching by combining bit-parallelism and suffix automata",ACM Journal. Experimental Algorithmics 1998.

[26] Hannu Peltola and Jorma Tarhio," Alternative Algorithms for Bit-Parallel String Matching", String Processing and Information Retrieval, Spire Springer, LNCS 2857, pp. 80-93, 2003.

[27] Longtao Hea, Binxing Fangaand Jie Sui," The wide window string matching algorithm" In the procd. Of Theoretical Computer Science of Elsevier, Vol. 332, pp. 391-404, 2005.

[28] L. Salmela, J. Tarhio, and J. Kytojoki, "Multi pattern string matching with q-grams", Journal of Experimental Algorithms, Volume 11, pp. 1-19, 2006.

[29] Branislav Durian, Jan Holub, Hannu Peltola and Jarma Tarhio,"Tuning BNDM with q-grams", In the proc. Of workshop on algorithm engineering and experiments, SIAM USA, pp. 29-37, 2009

[30] Changsheng Miao, Guiran Chang and Xingwei Wang," Filtering Based Multiple String Matching Algorithm Combining q-Grams and BNDM", In proc. Of Fourth International Conference on Genetic and Evolutionary Computing, 2010.