Variations of Wu-Manber String Matching Algorithm

Mahaveer Prasad Malav
Computer Science and Engineering
MANIT, Bhopal-462051
Madhya Pradesh, India

Prof. Akhtar Rasool
Computer Science and Engineering
MANIT, Bhopal-462051
Madhya Pradesh, India

Abstract **- Multi-pattern string matching is used in many research areas like security, data mining, machine learning, data analytics and search engines etc. Multi-pattern string matching covers the issue of finding the all possible occurrences of multiple patterns inside textual content in single pass. For multiple patterns matching large number of algorithms like Aho-Corasick, Wu-Manber (WM), Rabin-Karp and Shift-Or with Q-gram algorithms exists. Wu-Manber is one of the finest algorithm for string matching which have a flexibility to skip the unnecessary matching of text characters. After Wu-Manber a large number of modified Wu-Manber algorithms were proposed in recent times. In this paper we presented the basic Wu-Manber algorithm and their various variants. Here the comparative analysis of these algorithms is discussed briefly.**

Keywords- String Matching; WM; QWM; AFWM; HCWM; improved WM; BLAST algorithms

I. INTRODUCTION

String matching is used in almost all the software applications straddling from simple text editors to the complex Network Intrusion Detection System (NIDS)[1][2][3][4], Plagiarism Detection System[5], Digital Forensics tools[6], Text Mining tools[7] and many more. Due to string matching importance a large number of exact and approximate string matching algorithms were developed in last 60 years.

The various popular standard algorithms are existed for single pattern matching such as BM[8][9], KMP[2][10], Rabin Karp[11][12], BMH[9][13][14], BMHS[15], BMHS2[16], BMI[17], Improved BMHS[18] and BNDM[19] and TNDM[20]. Multi-pattern string matching algorithms were developed since 1970 such as Aho-Corasick [21][22][23][24], Commentz-Walter[22][25], Wu Manber[1][26][27][28], multi pattern string matching with q-grams[29][30] etc. These algorithms have many real world applications.

Multi-pattern string matching algorithms can be divided into two groups: automata-based algorithms and hashing based algorithms [33]. The main advantage of automata-based algorithms is to give similar in best, average and worst case matching performance. These algorithms scan the text in linear way and requires full scan of text. The negative aspect of automata-based algorithms is large memory space requirement because size of the trie grows enormously as the pattern set grows. These algorithms do not support scalability of pattern set means if new patterns added then recompilation of patterns is required [33].

In 1994 basic WM algorithm was proposed by Sun Wu and Udi Manber. The idea refers from two popular string matching algorithms i.e. Boyer-Moore (BM) [8] and Rabin-Karp (RK) [11]. The advantage of WM algorithm provides high average case performance and required small memory space. This algorithm introduces the concept of shifting of characters during scanning phase. Due to this shifting full text scanning is not required. WM algorithm have some issues like, it cannot handle more than few hundred patterns and only suitable for patterns which have similar length. When the length of patterns is short than searching time gets increased due to decrease in shifting of characters [31].

After WM algorithm many algorithms were proposed to improve the performance to overcome the limitations of WM. In 2006, Yang et al. proposed Quick Wu Manber (QWM) [26] algorithm. This algorithm uses the concept of Quick Search (QS) [32] algorithm. QWM introduces a new table named as head table. This head table contains the details of the first two characters of the patterns. Using this table QWM reaches the maximum shift distance in comparison to basic WM algorithm by using the mismatch details during the searching stage [26].

In 2008, Chen Zhen and Wu Di proposed Improved WM [33] algorithm, in which two Shift tables are introduced into basic Wu-Manber algorithm. As the number of patterns increases, values in single shift table are decrease. That results the costly exact string comparison to be performed with a higher probability. These two tables improve the possibilities of shifting text sliding window at each comparison [33].

In 2009 an aggressive algorithm for multiple pattern string matching called Quick Multiple Matching (QMM) [34], was proposed by Liuling Dai. QMM algorithm uses the concept of QS [32] algorithm and eliminates the dependency between Shift and Hash table in the Wu–Manber algorithm. This algorithm maximizes the shift distances in an aggressive manner by examining the character next to scan window [34].

Another improved algorithm called High Concurrence Wu Manber (HCWM) [35] was proposed in 2009 by Xiaoping

Chen et al. The key concept of the HCWM algorithm is how to deal with short patterns when mixed with long pattern. Divide all the patterns into different sets according to their length and process them in parallel [35].

In 2009 an improved WM algorithm based on address filtering named as AFWM [27] was proposed by Baojun Zhang et al. In AFWM Prefix table is utilized to filter the link list of possible matching patterns based on the address pointers of the patterns. According to the address pointers the patterns in the link list are sorted in ascending order. [27].

In 2013 Yoon-Ho et al. proposed the B-LAyered bad-character Shift Tables (BLAST) [36] for high-speed pattern matching algorithm. It uses the idea of B-layered bad-character SHIFT tables, i.e. multiple shift tables for single character instead of block of character is used to improve the maximum shift distance during the scanning phase in text [36].

TABLE 1. History of various improved WM algorithms

S.No.	Algorithm	Year	Authors
1	QWM	2006	Yang et al.
2	Improved WM	2008	Chen Zhen et al.
3	QMM	2009	Liuling Dai
4	HCWM	2009	Xiaoping Chen et al.
5	AFWM	2009	Baojun Zhang et al.
6	BLAST	2013	Yoon-Ho et al

II. BASIC WM ALGORITHM

WM algorithm advances the multi-pattern string matching algorithms by introducing a new concept of shifting (skipping) the unnecessary matched characters in searching phase based on block of characters. This skipping concept improves the speed of searching and it is faster than the previous automata based algorithms. The algorithm can be divided into two phases; 1) Pre-processing phase, 2) Scanning phase.

1) Pre-processing phase

In pre-processing phase three tables named: Shift table, Hash table, and Prefix table are built by processing the set of patterns. Firstly calculate 'm', which is equal to the minimum pattern length. Now the next step is to construct three tables by considering only first 'm' characters of each pattern. The Shift table is similar to the regular Shift table in a BM algorithm, but not exactly the same. In BM shift table is based on the single character of the patterns whereas in WM it is based on the block of characters. It is used to determine how many characters in the text can be shifted (skipped) at the time of text scanning. When there is a possible match occurs in shift table, the Hash and Prefix tables are used. Hash table contains a pointer to the list of patterns whose suffix is same whereas Prefix table contains a hash value of prefix of each pattern. These tables are used to find the actual pattern for the match [31]. The Actual process of pre-processing, creating three tables is described below:

Shift Table: Shift table contains all possible strings of size B; in practice, most of the times the value of B is either 2 or 3. The value in shift table is the largest possible safe value for shift. Initialize Shift table with the value m-B+1 i.e. the maximum shift distance when B character of text are not appears in any of the pattern. Now update the Shift table by *minimum (current value, m-q)* i.e. B characters appears in some patterns, In this case, find the rightmost occurrence of these characters in any of the patterns; let's assume that it is at position q in a pattern than store m - q in Shift table. Actual implementation of Shift table is described below:

Assume we have the following patterns P= {alive, annual, announce}. Let block length (B) = 2, minimum pattern length (m) = 5. Than the shift table is constructed as follows:

TABLE 2. Shift table for the example of WM

al	li	iv	ve	an	nn	nu	no	ua	ou	*
3	2	1	0	3	2	1	1	0	0	4

Hash Table: Hash table is built by mapping the last B characters (considering only first 'm' characters) of all patterns. In order to avoid comparing the each pattern from the pattern set with the text, we make use of hashing method that reduces the number of comparisons. Hash table is computed using exact same integer as in Shift table as an index. Each entry in the Hash table contains a pointer to a list of patterns whose last B characters are same. Actual implementation of Hash table for above example is described below:

TABLE 3. Hash table for the example of WM

ve	ua	ou	*
1	2	3	null

Prefix Table: The Prefix table is built by mapping the first B' characters of all patterns (Generally B' = 2 is a good value). Use of Prefix table is to filter patterns whose suffix is the same but whose prefix is different. The Hash table not only contains, for each suffix, the list of all patterns with this suffix, but it also contains an index to the Prefix table.

2) Scanning phase

First of all put a window of size 'm' in the text as current scanning text and calculate a hash value index of last B (generally B=2 or 3 is a good value) characters in that window. Now Check the value for that index in Shift table; if the value is greater than zero, then right shift the text window by the value stored in Shift table and continue the scanning; if value in shift table is zero in that case, same index in Hash table gives the list of patterns whose suffix are same. Calculate the hash value of the prefix of the text window by moving m-B' characters left from current position; call it window_prefix. Now check for each pattern from the pattern list, if their value in Prefix table is matched with window_prefix. If they are equal, check the actual pattern against the text directly. If any of the patterns matched mark its occurrence and shift the text window by one. Continue the whole process till the end of text. Example of basic WM

algorithm for the text T = "strcmatecadannualho" is described below:

The algorithm proceeds as follows:

step	Text	shift	output
1	strcmatecadannualho	4	
2	strcmatecadannualho	4	
3	strcmatecadannualho	3	
4	strcmatecadannualho	0	annual
5	strcmatecadannualho	4	

Fig. 1. Wu Manber (WM) searching process.

The advantage of WM algorithm as compare to automata based algorithms is, it has less number of comparisons in text scanning and also no need for additional space for data structure and no need for organizing the data structure, e.g. sorting.

III. IMPROVEMENT OF WM ALGORITHM

A. *Quick Wu Manber(QWM) Algorithm*

In Wu-Manber algorithm, the maximum shift distance in Shift table is m-B+1 which is improved in QWM algorithm. The limitation of WM algorithm is, during the searching process when value in the Shift table is equal to zero, the shift distance is always equal to 1 whether there is full match or not. QWM algorithm uses the concept of QS [32] algorithm, reached the maximum shift distance m by using the mismatch details during the searching stage [26].

In pre-processing stage, in addition to the Shift table, the Hash table, and the Prefix table, one more table is introduced named as Head table, which contains the information about first two characters of the patterns. The Prefix table and the Hash table is calculated same as the Wu-Manber algorithm. The Shift table is little bit changed as store m (instead of m-B+1) during initialization of Shift table remaining process of creating shift table is same as WM algorithm [26].

The use of Head table is to find whether the first two characters in the text which we are currently scanning is the prefix of any pattern. Initialize Head table with the value 0 that means the first two characters does not appear in any pattern. Now update Head table with value 1, for first two characters who appears in some patterns as prefix [26].

In the searching stage, QWM uses the concept of QS algorithm in which scanning of the text is done from left to right. The first step is to check whether the current text is the prefix of any pattern. If it is, then follow the method of Wu-Manber algorithm. Whether there is a complete match or not, the shift distance always equal to the value in Shift table. Example of QWM algorithm is described below:

For pattern set {search, hear, arch, chart} and text string "strcmatecadnsearchof", for B=2 and m=5. Assume we are given the following tables:

TABLE 4. Shift table for the example of QWM

ar	ch	ea	ha	he	rc	se	*
1	1	2	2	3	1	3	4

TABLE 5. Head table for the example of QWM

se	he	ar	ch	*
1	1	1	1	0

Then the algorithm proceeds as follows:

step	Text	head	shift	output
1	strcmatecadnsearchof	0	4	
2	strcmatecadnsearchof	0	4	
3	strcmatecadnsearchof	0	4	
4	strcmatecadnsearchof	1	0	search
5	strcmatecadnsearchof	0	1	

Fig. 2. Quick Wu Manber (QWM) searching process.

The advantages of QWM algorithm over basic WM algorithm is, it reaches the maximum shift distance m instead of m-B+1 by using the idea of QS algorithm. But the limitations of QWM algorithm is, in each scanning step two hash calculations(for Head and Shift table) are performed to get shift distance whereas in basic WM algorithm only one needed. And also the complex of Head table computation depends on the size of pattern set. It requires more memory as compare to the basic Wu-manber algorithm because in pre-processing four tables are created instead of three.

B. *Improved WM Algorithm*

A second Boyer-Moore Shift table into basic Wu-Manber algorithm is introduced in this paper, so that the possibilities of shifting text sliding window at each comparison can be improved; by using second shift table expensive exact string comparison is avoided quite often, which leads this algorithm towards a faster multi-pattern matching algorithm. As compare to the basic Wu-Manber algorithm, the improved Wu-Manber algorithm is much faster and suitable for gateway-based online content scanning scenarios [33].

The improved Wu-Manber algorithm introduced with three differences as compare to the basic Wu-Manber algorithm: 1) to better the quality of the Boyer-moore like Shift table, for each original pattern a pattern representative (rarest substring with fixed length) is chosen; 2) to increase the possibilities of shifting text sliding window, a second Boyer-moore Shift table is computed; 3) to design a balanced Hash table for improving the scanning speed of possible matching patterns a simple hash function with good randomness property is crafted [33].

The advantages of improved algorithm over the previous explained algorithms is that the text scanning speed of the improved Wu-Manber algorithm is faster because with the use of second shift table exact string comparison is avoided most of the time. This algorithm have some limitations like calculation of two shift tables is not a good idea in pre-processing stage and when the number of patterns reaches

40,000 or more, the introduced extra shift table still cannot improve the performance.

C. *Quick Multiple Matching (QMM) Algorithm*

The limitation of basic WM algorithm is that when the value in Shift table is zero then we go to check for the Hash table, in other words when the Hash table is not null then there must be larger (greater than zero) value in the shift table. This function overlap between Hash table and Shift table is eliminated in this algorithm [34].

QMM algorithm modifies the basic WM algorithm by introducing character next to scan window in Shift table. It uses the idea of QS algorithm as Shift table is only used to get shift distance. Shift table have the entry by hashing the last char of scan window and next to last character of scan window. The Hash table is used for finding the possible matching patterns list whether it is null or not. So the shift table is not depended on the possible matching, the maximum shift distance (m instead of m-B+1) is achieved using this concept [34]. Example of QMM algorithm is described below:

Assume we have to find patterns P= {alive, annual, announce} inside the text T = "strcmatecadannualho". Let block length (B) = 2, minimum pattern length (m) = 5 and assume we are given the following tables:

TABLE 6. Shift table for the example of QMM

al	li	iv	ve	an	nn	nu	no	ua	ou	*
4	3	2	1	4	3	2	2	1	1	5

TABLE 7. Hash table for the example of QMM

ve	ua	ou	*
1	2	3	null

Then the algorithm proceeds as follows:

step	Text	hash	shift	output
1	strcmatecadannualho	null	5	
2	strcmatecadannualho	null	5	
3	strcmatecadannualho	null	1	
4	strcmatecadannualho	2	4	annual

Fig. 3. Quick Multiple Matching (QMM) searching process.

The advantages of QMM algorithm is, the maximum shift distance m is achieved in Shift table as it is independent on the possible match of patterns, its only use is to get the shift distance and also the functional overlap between Hash table and Shift table is eliminated. But the algorithm has the limitation that at each scanning step both shift and Hash table is checked to get shift distance and to find possible match, whereas in basic WM algorithm does not check Hash table until value in shift table gets 0.

BLAST algorithm is based on multiple layered shift tables with single character search at a time. BLAST algorithm overcomes the limitations of multi character shift table (as it has average shift values in typical search) used in various WM algorithms. The shift tables in BLAST algorithm is based on single character, so it takes less memory for shift

D. *High Concurrence Wu Manber (HCWM) Algorithm*

The performance of Wu-Manber algorithm is greatly affected when short patterns are mixed with long patterns. In order to solve this problem, High Concurrence Wu-Manber Algorithm is proposed. It uses the concept of dividing all the patterns into different sets according to their length and for each set, independent data structures are established and different process methods are used in parallel to obtain high concurrency when searching patterns in text, because there are few resources shared among these sets. The search operation is executed concurrently because each pattern set has independent data structures, and the common text file is read only. Multiple threads are used for the pattern matching [35].

When short patterns and long patterns mix together, the performance of HCWM is better than WM obviously. The reason is that HCWM processes long patterns and short patterns in different ways. But the limitations of this algorithm are pre-processing or organizing the data structure takes time, because it divides all the patterns into different sets.

E. *Addressing Filtering Wu Manber (AFWM) Algorithm*

In this paper a new approach to improve the basic Wu-Manber algorithm is explained which is based on address filtering. The basic Wu-Manber algorithm has the following limitations. 1) There are redundant information and operations. 2) The Prefix table for filter pattern is constructed, but is hardly used. 3) Need to traverse the whole link list. These limitations affect the performance of algorithm. Address filtering based search method is introduced to overcome these limitations [27].

For filter the patterns Prefix table is used. Address filtering based search method avoids traversing the whole link list with the help of Prefix table. AFWM algorithm has only difference with the basic Wu-Manber algorithm is in the prefix link list; all the same prefix patterns will be sorted in ascending order according to the address pointers of the patterns. So, we only need to match the patterns whose address pointers locate in the range of Hash table. In this way the search process of improved algorithm can be finished faster than the basic Wu-Manber algorithm [27].

The advantages of address filtering algorithm is, it avoids traversing the whole link list and with the help of address pointer is it uses the Prefix table sufficiently. There are some limitations in this algorithm such as improved effect of AFWM is sensitive to the number of patterns, when the number of patterns increases; the effect of AFWM is more obvious. The reason is that as the number of patterns increasing, the length of link list will become longer.

F. *B-layered bad-character SHIFT tables (BLAST)Algorithm*

table as compare to other WM algorithms. This algorithm also resolves the performance degradation issue, when the frequency of last character occurrence is higher in single character table [36].

BLAST algorithm scans the text by using B-layered bad-character SHIFT tables (where B is the size of search unit). Suppose the value of B is 2 in that case two shift tables are created in preprocessing stage. In this way multiple layers are created and entries in each shift table is based it's above layer shift table. In the scanning phase if all shift tables gives the value 0 in that case we check the Hash table for possible match otherwise shift the text according to value of particular layer SHIFT table [36]. Example of QMM algorithm is described below:

Assume we have to find patterns P= {alive, ping} inside the text T = "strkalliepingho". Let block length B=2, minimum pattern length (m) = 4 and assume we are given the following tables:

TABLE 8. Two Layered SHIFT table for BLAST

characters	a	l	i	v	p	n	g	*
$SHIFT_{L0}$	3	2	1	0	3	1	0	4
$SHIFT_{L1}$	3	2	0	0	3	0	0	4

Then the algorithm proceeds as follows:

step	Text	$SHIFT_{L0}$	$SHIFT_{L1}$	output
1	strkalliepingho	4		
2	strkalliepingho	1		
3	strkalliepingho	4		
4	strkalliepingho	0	0	ping
5	strkalliepingho	4		

Fig. 4. B-layered bad-character SHIFT tables (BLAST) searching process.

The advantages of BLAST algorithm is maximum shift distance m is achieved with the use of single character shift tables and also the memory size of shift table is reduced as compare to multi character shift table used in the basic WM algorithm. But the performance of BLAST algorithm decreases, if every character in the rightmost position of the patterns is present, because BLAST algorithm's SHIFT tables are based on a single character search technique, so the Hash table will be compared for every character in the text.

TABLE 9. A Comparative Analysis of Various WM algorithm, where 'm' is minimum length pattern and 'B' is block characters length

S.N0.	Algorithm	Maximum Shift Distance	Number of Pre-processing Tables Required	Key Ideas
1	WM	m-B+1	3(shift, hash, prefix)	Uses idea of Shift table is for escaping the characters in text, hash table for list of possible match and prefix table for filtering the patterns.
2	QWM	m	4(head, shift, hash, prefix)	Head table used as prefix matching (scanning text from left to right).
3	Improved WM	m-B+1	4(shift1, shift2, hash, prefix)	Two Shift tables are used for finding the possible match.
4	QMM	m	3(shift, hash, prefix)	Use Hash table and Shift table independently.
5	HCWM	m-B+1	3(shift, hash, prefix)	Divide pattern set into different groups according to pattern length and process them in parallel.
6	AFWM	m-B+1	3(shift, hash, prefix)	Address pointers are used to sort the pattern list.
7	BLAST	m	4($shift_{L0}$, $shift_{L1}$, hash, prefix)	Multiple layers of shift tables are used.

IV. CONCLUSION

In this paper core ideas of basic Wu Manber (WM) and its various improved algorithms are discussed. The performance of all the above described algorithms is depends on shift table and the hash function used in these algorithms. Some algorithms tries to improve the effect of short patterns when mixed with long patterns and some algorithm improves the maximum shift distance in the shift table. QWM and QMM algorithm uses the idea of QS algorithm. Improved WM and BLAST algorithm uses multiple shift tables. HCWM is based on parallel processing of pattern set and AFWM is based on Address filtering of patterns. All the described algorithms can be used in various fields, such as text retrieval, plagiarism detection system, intrusion detection, network content analysis etc.

REFERENCES

[1]. Zhengqiang," An improved multiple patterns matching algorithm for intrusion detection", In the proc. of IEEE International Conference on Intelligent Computing and Intelligent Systems (ICIS),Volume 2, pp. 124-127, 29-31 Oct. 2010.

[2]. 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.

[3]. Byung-joo Kim, Kim, "Kernel based intrusion detection system", In the proc. of Fourth Annual ACIS International Conference on Computer and Information Science, pp. 13-18, 2005.
[4]. Cheng Ke-qin , Deng Lin , Wang Hui ,"An improved multi-pattern matching algorithms in intrusion detection" In the proc. Of 2013 Fifth International Conference on Measuring Technology and Mechatronics Automation (ICMTMA), Hong Kong, pp. 203 – 205, 16-17 Jan. 2013.
[5]. Alzahrani S.M. Salim N. and Abraham A., "Understanding Plagiarism Linguistic Patterns, Textual Features, and Detection Methods", In the proc. of IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews", Vol. 42, Issue 2, pp. 133-149, march 2012.
[6]. Marturana, F, Gianluigi Me and Tacconi, S,"A Case Study on Digital Forensics in the Cloud", In the proc. of International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (Cyber C), pp. 111-116, 10-12 Oct. 2012.
[7]. 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, 15-19 Dec. 2008.
[8]. R. S. Boyer and J. S. Moore, "A Fast String Searching Algorithm", Communications of the ACM, Vol. 20(10), pp.762-772, 1977.
[9]. 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.
[10]. Donald E. Knuth, James H. Morris, Jr. and Vaughan R. Pratt, "Fast pattern matching in strings", In proc. of SIAM Journal on Computing, vol. 6 (2), pp. 323–350, 1977.
[11]. R. M. Karp and M. O. Rabin, "Efficient randomized pattern-matching algorithms," In: (2nd ed.), Tech. Rept. 31-81, Aiken Computer Lab, Harvard University, Cambridge, MA, 1981.
[12]. Brodanac P., Budin, L. and Jakobovic," Parallelized Rabin-Karp method for exact string matching", In the proc. 33rd International Conference on of Information Technology Interfaces (ITI), pp. 585-590, 27-30 June 2011.
[13]. 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.
[14]. R. N. Horspool ,"Practical fast searching in strings", In the proc. of Software - Practice & Experience, Vol. 10, pp. 501–506, 1980.
[15]. 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.
[16]. 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.
[17]. 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.
[18]. 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.
[19]. G. Navarro and M. Raffinot, "Fast and flexible string matching by combining bit-parallelism and suffix automata", In the proc. of ACM J. Experimental Algorithmic (JEA), Vol. 5 (4), 2000.
[20]. Hannu Peltola and Jorma Tarhio, "Alternative Algorithms for Bit-Parallel String Matching, String Processing and Information Retrieval", Lecture Notes in Computer Science, Springer, Vol. 2857, PP 80-93, 2003.
[21]. Aho and M. Corasick, "Efficient string matching: An aid to bibliographic search", Communications of the ACM, vol. 18, pp. 333–340, June 1975.
[22]. 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.
[23]. 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.
[24]. Xinyan Zha and Sartaj Sahni, "GPU-to-GPU and Host-to-Host Multi-pattern String Matching on a GPU", In the proc. of IEEE Transactions on computers, Vol. 62, NO. 6, June 2013.
[25]. 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, Sept. 30 2011-Oct. 2 2011.
[26]. 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.
[27]. 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.
[28]. 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.
[29]. Fang Xiangyan, Xiong Tinggang, Ding Yidong and Youguang, "The research and improving for multi-pattern string matching algorithm", In the proc. of 2010 IEEE International Conference on Intelligent Computing and Intelligent Systems (ICIS), Vol. 1, pp. 266-270, Oct. 2010.
[30]. Changsheng Miao, Guiran Chang and Xingwei Wang,"Filtering Based Multiple String Matching Algorithm Combining q-Grams and BNDM", In the proc. of Fourth International Conference on Genetic and Evolutionary Computing (ICGEC), pp. 582-585, 13-15 Dec 2010.
[31]. L. Salmela, J. Tarhio, and J. Kytojoki, "Multi pattern string matching with q-grams", In the proc. of Journal of Experimental Algorithmic, Volume 11, pp. 1-19, 2006.
[32]. Wu S. and U.Manber, "A Fast Algorithm for Multi-Pattern Searching," Technical Report TR-94-17 Department of Computer Science, University of Arizona, Tucson, AZ (May 1994).
[33]. D.M. Sunday, "A Very Fast Substring Search Algorithm", Communications of the ACM, Vol. 33, 8, pp. 132-142, 1990.
[34]. Chen Zhen and Wu Di, "Improving Wu-Manber: A Multi-pattern Matching Algorithm", In the proc. of 2008 IEEE International Conference on Networking, Sensing and control (ICNSC), pp. 812 – 817, 6-8 April 2008.
[35]. Liuling Dai,"An aggressive algorithm for multiple string matching" Information Processing Letters, Volume 109, pp. 553–559, May 2009.
[36]. Baojun Zhang, Xiaoping Chen, Xuezeng Pan, and Zhaohui Wu "High concurrence Wu-Manber Multiple Patterns Matching Algorithm", Proceedings of the International Symposium on Information Proces, p.404,August 2009.
[37]. Yoon-Ho, Seung-Woo,"BLAST: B-LAyered bad-character SHIFT tables for high-speed pattern matching", Journal of Information Security, Institution of Engineering and Technology (IET), Volume 7, pp.195-202,sept.2013.