A COMPARATIVE ANALYSIS OF MULTIPLE PATTERN MATCHING ALGORITHMS

Renuka S. Matkar[1], Yogeshwar. M. Kurwade[2], Dr. Vilas M. Thakare[3]

[1,2,3]SGBAU, Amravati, Maharashtra (India)

ABSTRACT

Multiple pattern matching algorithm is a type of string searching algorithm. From a given finite set of strings (patterns), it can locate all patterns' locations simultaneously in a text string. Pattern matching plays an important role in many computer related fields, such as information retrieval, intrusion detection, data compression, content filtering, gene sequence comparison and computer virus signature matching. It is a basic problem in computer science. There are many multiple pattern matching algorithms but to provide the better results, a comparative study of all of them is necessary. The comparative analysis gives us the clear idea to choose the best algorithm for pattern matching problem. In this paper, the comparative analysis of the most widely used multiple pattern matching algorithms is given. This paper analyzes and discusses about the widely used multiple string pattern matching algorithms and also proposes a new multi pattern matching algorithm using frequently used groups. The mostly used algorithms discussed in this paper are: the Aho-Corasick algorithm, the Commentz-Walter algorithm, and the Wu-Manber algorithm. The comprehensive analysis and discussion also helps for the better understanding of the pattern matching problem. A theoretical and experimental result of the algorithm is presented in this paper. The paper is concluded with the solution for the multi-pattern matching problem with a new algorithm.

Index Terms: Algorithms, Frequently Used Groups, Pattern Matching, String Matching, String Searching

I. INTRODUCTION

String pattern matching or searching is the act of checking for the presence of the constituents of a given pattern in a given text where the pattern and the text are strings over some alphabet. Pattern matching plays an important role in many computer related fields, such as information retrieval, intrusion detection, data compression, content filtering, gene sequence comparison and computer virus signature matching. [1][2][3]. Pattern matching algorithms belong to computationally intensive algorithms. It can be classified into single pattern matching and multi-pattern matching according to the number of matching patterns. Single-pattern matching came first, and there are some classic algorithms such as the Knuth-Morris-Pratt (KMP) algorithm and the Boyer-Moore (BM) algorithm, which offer some lessons and inspirations for the development of later multi-pattern matching algorithm [1]. Multi pattern matching algorithm can find all occurrences of multiple patterns with only once scanning. It is more complex to implement but a wider range of applications than single-pattern matching algorithm. In a multiple pattern matching, from a given finite set of strings (patterns), it can locate all patterns' locations simultaneously in a text string. Multiple pattern matching algorithms can be used in data mining area to find selected interesting patterns, security area to detect certain suspicious keywords or even biological field for DNA searching [2]. Pattern matching is one of the major issues in the area of computational biology. Biologists search database for important information in different directions. Pattern matching will continue to grow and need changes from time to time. The analysis of Protein and DNA sequence data has been one of the most active research areas in the field of computational molecular biology. Pattern Discovery is one of the fundamental problems in Bioinformatics. It can be used in Protein structure and function prediction [3]. The pattern matching algorithm is the core algorithm of the entire anti-virus software. Combining the advantages of fast calculation of hash function and parallel pattern matching of automata, it has significant performance advantages in the circumstance of virus signature matching [4]. Multiple string pattern matching problems has been a topic of intensive research that has resulted in several approaches for the solution such as multiple keyword generalization of Boyer Moore algorithm, Boyer-Moore-Horspool algorithm, Aho-Corasick algorithm, Commentz-Walter algorithm, Fan-Su algorithm, Wu-Manber algorithm, and Set Backward Oracle Matching (SBOM) algorithm. The most popular and widely used solutions for the multi-pattern matching problem are: Aho-Corasick algorithm, Commentz-Walter algorithm, and Wu-Manber Algorithm [5].

This paper mainly presents the analysis of mostly used algorithms for multiple string pattern matching problems: the Aho-Corasick algorithm, the Commentz-Walter algorithm, and the Wu-Manber algorithm. Experimental results of these algorithms are included for the analysis and discussion about multiple pattern matching problems. This paper also discusses the main theoretical results for each of the algorithm. The performance of each algorithm is shown against the length of pattern and the number of pattern in a pattern set. This paper covers the comprehensive analysis and discussion of these selected algorithms as a state-of-the-art along with some experimental results. A comprehensive study on all the existing algorithms of multiple pattern matching problems is a very demanding material in the research area of multiple pattern matching problems. In short, this paper provides the best solution on any type of pattern matching problem in easier manner.

II. BACKGROUND

Along with the rapid development of computer technology, people's lives are increasingly dependent on computers. The rapid development of the Internet increases the freedom of application. But at the same time, because of its inherent openness, universality and freedom, it requires a higher demand of information security. Pattern matching algorithms can be classified into single pattern matching and multi-pattern matching according to the number of matching patterns. Single-pattern matching appeared first, and there are some classic algorithm such as the Knuth-Morris-Pratt (KMP) algorithm and the Boyer-Moore (BM) algorithm, which offer some lessons and inspirations for the development of later multi-pattern matching algorithm. The basic need of pattern matching is proposed in [1].

Multiple pattern matching problem is a basic problem in computer science. As a solution, many algorithms are generated. Aho-Corasick algorithm, a variant of the Knuth-Morris-Pratt algorithm, was the first algorithm to solve the multiple string pattern matching problems in linear time based on automata approach. Commentz-Walter presented an algorithm for the multi-pattern matching problem that combines the Boyer-Moore technique with the Aho-Corasick algorithm. Commentz-Walter combines the filtering functions of the single pattern matching Boyer-Moore algorithm and a suffix automaton to search for the occurrence of multiple patterns in an input string. Wu-Manber algorithm is a simple variant of the Boyer-Moore algorithm that uses the bad character shift for multiple pattern matching. Mostly used algorithms for multiple pattern matching are discussed in [2].

Brute-force algorithm performs a checking, at all positions in the text between 0 and n-m, whether an occurrence of the pattern starts there or not. The Boyer-Moore Algorithm works with a backward approach, the target string is aligned with the start of the check string, and the last character of the target string is checked against the corresponding character in the check string. The Index based Pattern Matching using Multithreading method performs pre-processing to get the index of the first character of the pattern in the given text. By using this index as the starting point of matching, it compares the Text contents from the defined point with the pattern contents. An improved approach for multi pattern matching using index base method is proposed in [3].

IAC: an improved multi-pattern matching algorithm based on AC algorithm. AC algorithm is a string searching algorithm invented by Alfred V. Aho and Margaret J. Corasick. AC algorithm constructs a finite state machine consisting of Goto, Failure and Output Functions. Goto Function is graph (tree). An improved approach for existing AC algorithm is presented in [4].

Solutions to different problems were proposed by Commentz Walter in 1979 [5] and Navarro Raffinot [5] in 2002. Both offered sub linear time solutions for single pattern matching. Recently, Khancome and Boonjing proposed searching techniques which used shift table algorithms at their core. Now-a-days, clustering and hash based technology to prevent intrusion detection system is used on widespread. The recent applications of pattern matching are proposed in [5].

This paper is organized as follows: Section I describes a brief introduction to multiple pattern matching. Section II briefly describes the background of multiple pattern matching algorithms specifically Aho-Corasick, Commentz-Walter, and the Wu-Manber algorithm. Section III outline the experiment methodology for new multi pattern matching algorithm using frequently used groups, Section IV presents the experimental results on the multiple pattern matching algorithms, Section V presents the analysis and discussion on pattern matching problem based on the experimental results and existing works, and Section VI gives the conclusion of this paper.

III. PREVIOUS WORK DONE

S.Nirmala Devi et al. (2012) [4] and Dr.S.P Rajagopalan et al. (2012) [4] proposed the Index based Pattern Matching using Multithreading method performs pre-processing to get the index of the first character of the pattern in the given text. By using this index as the starting point of matching, it compares the Text contents from the defined point with the pattern contents. Raju Bhukya et al. (2012) [4] and DVLN Somayajulu et al. (2011) [4] proposed IBSPC. In IBSPC indexes has been used for the DNA sequence. After creating the index the algorithm will search for the pattern in the string using the index of least occurring character in the string.

Raju Bhukya et al. (2010) [3] and DVLN Somayajulu et al. (2010) [3] proposed IFBMPM. In IFBMPM to search some pattern P in text S, it start the search from the indexes stored in the row of index table which corresponds to the first character of the pattern P. If any character mismatches in its position, we skip the search and go for the next index which corresponds to the first character of the pattern P according to the indexes stored in index table for matching.

ISBN: 978-93-86171-47-4

Reverse Colussi et.al. (2009) [1] proposed RC algorithm in which comparisons are done in specific order given by the preprocessed phase. The time complexity of preprocessing phase is O(m2) and searching phase is O(n). When several pattern strings need to match, using Single pattern matching has low efficiency. Corasick M.J et.al (2008) [1] proposed many pattern matching algorithm with high efficiency to solve this problem, which is called for short AC algorithm. In the preprocess stage, AC algorithm form several pattern strings waiting for matching, according to their features into Tree finite state automata, and decide the next situation according to matching characters.

Y. H. Cho et. al. (2008) [2] proposed a hash-based pattern matching co-processor where memory is used to store the list of substrings and the state transitions. Dharmapurikar *et al.* (2007) [2] proposed a pattern matching algorithm which modifies the AC algorithm to consider multiple characters at a time. Furthermore, the content addressable memories (CAM) is also widely used for string matching because it can match the entire pattern at once when the pattern is shifted past the CAM.

The Aho–Corasick (AC) algorithm (1975) [5] is the most popular algorithm which allows for matching multiple string patterns. Aldwairi *et al.* (2005) [5] proposed a configurable string matching accelerator based on a memory implementation of the AC FSM.

IV. EXISTING METHODOLOGY

Multiple Pattern Matching Algorithms

A. Aho-Corasick algorithm

Aho-Corasick algorithm is a variant of the Knuth-Morris-Pratt algorithm. It was the first algorithm to solve the multiple string pattern matching problems in linear time based on automata approach. Aho-Corasick algorithm serves as a basis for the UNIX tool fgrep. It consists of two parts. In the first part a finite state pattern matching machine is constructed from the set of keywords and in the second part, the text string as input is applied to the pattern matching machine. The machine gives signals whenever it finds a match for a keyword (pattern). The pattern matching machine consists of a set of states and each state is represented by a number. The behavior of the pattern matching machine is explained by following three functions: a goto function g, a failure function f, and an output function output [1].

The function of goto function g is that it maps a pair consisting of a state and an input symbol into a state or message fail. The failure function f maps a state into a state. The failure function is visited whenever the goto function reports fail. The output function of certain states indicates that a set of keywords has been found. The construction of Aho-Corasick automaton machine takes running time in the sum of the lengths of all keywords linearly. This involves building of the keyword tree for the set of pattern and then converting the tree to an automaton also called as pattern matching machine by defining the functions g (goto function), f (failure function), and output function for naming states with the keywords matched. The memory or space requirements of the Aho-Corasick algorithm can be very large depending on the pattern set and also the length of each pattern in a pattern set. The matching process simply steps through the input characters one at a time and then checks if there is any matching. Each step in pattern matching machine happens in constant time. The Aho-Corasick matcher always operates in O(n) running time [1].

 ISBN: 978-93-86171-47-4

The good suffix rule is used as a formula in AC algorithm.

I) $shift(j) = min\,(s|(p(j+1 \ldots m) = p(j-s+1 \ldots m-s)))$

Where, assuming shift j is the distance which P skips to right, m is the length of pattern string P, j is the current matched character position, s is the distance between and t or the distance between X and p".

B. Commentz-Walter algorithm

From combination of the Boyer-Moore technique with the Aho-Corasick algorithm, Commentz-Walter presented the Commentz-Walter algorithm. It combines the filtering functions of the single pattern matching Boyer-Moore algorithm and a suffix automaton to search for the occurrences of multiple patterns in an input string. Commentz-Walter used a tree in his algorithm which is similar to that of Aho-Corasick' pattern matching machine but is created from the reversed patterns. The Commentz-Walter algorithm consists of two phases: 1) pre-processing phase and matching phase. The pre-processing phase of algorithm is responsible for creating a pattern tree by using the reversed pattern. The matching phase of the Commentz-Walter algorithm is the combination of two ideas. One is from the idea Aho-Corasick' finite automata technique and another one is from the Boyer-Moore shifting technique (in right-to-left matching). In this algorithm a match is conducted by scanning backwards through the input string. When the mismatch occurs, some number of characters about the input string is known and this information is then used as an index. This index is used in a pre-computed table to determine a distance which later helps to shifting before occuring the next match attempt [2].

For calculating the shift distance, bad character of shift left rule is used as shown in below:

$$Bad\ char\ of\ shift\ l(a,b,c) = min \begin{cases} m+1 \\ m+2 \\ m+3 \end{cases}$$

Where, min(m+1), (m+2), (m+3) are the minimum shift distances.

C. Wu-Manber algorithm

Wu-Manber algorithm is a simple variant of the Boyer-Moore algorithm. It uses the bad character shift rule for multiple pattern matching. After making a UNIX based tool agrep, this algorithm was proposed in [3]. It was used for searching many patterns in files. To improve the performance, a unique idea was created, that is, their algorithms looks at block of text instead of a single character. So, they consider both pattern and text as blocks of size B instead of single characters. Hash value is also known as a "message digest". The mathematical expression:

$h = H(M)$

Where, H() is a one-way hash function, M is an input with arbitrary length, and h is the fixed length hash value.

D. Boyer-Moore algorithm

These are some principles to realize the BM algorithm: 1) at the beginning of matching, align pattern strings P and text T from left to right, but the matching operation starts from right to left.

 ISBN: 978-93-86171-47-4

2) If the character and position in P matches with the character in text T, T and P will move a position toward left at the same time and then make comparison.

3) If the matching fails, two offset functions Badchar and Goodsuffix in preprocessing will work out the distance in which pattern string P moves toward right, and align T and P again to match. Here are the specific definitions of functions Badchar and Goodsuffix [4].

a) Use Badchar to move: Work out the deviant of every character in the T character collection. If character C in T appears many times in P, the last position can be used to work out the deviant, the mathematical formula is as follows:

$$\text{Badchar(c)} = \begin{cases} m, \text{any } c \text{ not in } P_m \text{ and } c \neq P_j (1 \leq j \leq m-1) \\ m - j, \text{here } j = MAX\{j | c = pj, 1 \leq j \leq m-1\}, \\ \qquad c \in P \end{cases}$$

E. Hash Algorithm

Hash algorithm converts the input of any length to a smaller output of fixed length through the hash function, and the output value is called the hash value. The hash value is a unique and extremely compact numerical representation of a piece of data. This conversion is a compressed mapping, that is to say, the space occupied by hash value is much smaller than that of the input. It's statistically impossible to produce two inputs with the same hash value.

That is to say, given M, it's computationally difficult to find M' to meet $H(M) = H(M')$ which we called weak collision resistant. Also, it's computationally difficult to find any pair of

M and M' to meet

$$H(M) = H(M')$$

This is called as strong collision-resistant.

V. ANALYSIS AND DISCUSSION

A linear time algorithm for multiple patterns matching problem which is proposed by Aho and Corasick is optimal in worst case but Boyer and Moore demonstrated an algorithm where they showed that it is possible to skip a large portion of the text while searching for certain patterns. Eventually, the approach by Boyer-Moore is working faster than linear algorithm in the average case [1].

The Commentz-Walter algorithm is the combination of idea of Boyer and Moore technique with Aho-Corasick algorithm for multiple pattern matching problems which is greatly faster than the Aho-Corasick algorithm in practice. It uses the idea of Boyer Moore technique to skip a large portion of the text while searching and as a result it leads to faster than linear time algorithms in the average case [2].

Wu-Manber algorithm is the most efficient algorithm under some situations such as long random patterns, low matching rate, and low memory requirement. The performance of Commentz-Walter algorithms declined with increasing number of pattern in a pattern set (pattern set size). The performance of Commentz-Walter algorithm improved approximately linearly with increasing length of the shortest keyword/ pattern in the pattern set [3].

There is another algorithm proposed by Baeza-Yates which also combines the idea of Boyer-Moore-Horspool algorithm (which is a slight variation of the classical Boyer-Moore algorithm) with the Aho-Corasick algorithm. Where, However, Aho-Corasick performance does not suffer great loss when comparing with others because it is a linear time searching algorithm in worst case. Independent from the pattern set size, searching time complexity for Aho-Corasick algorithm is O(n) but when pattern set size increase, the memory consumption increased drastically and also the time consumption increased [4].

The Aho-Corasick and Commentz-Walter algorithms consume lots of memory because both these algorithms use the automata data structure in the preprocessing stage where Wu-Manber algorithm consumes much less memory than these two algorithms [5].

Table 1: Comparative analysis of the existing methodologies

Methodology/ Algorithm	Advantages	Disadvantages
1. Aho-Corasick algorithm	1. As it is a native algorithm, it used as a linear solution to many problems. 2. Use of simpler approach	1. It operates in O(n) times, so more execution time than other approaches. 2. More no. of comparisons.
2. Commentz-Walter algorithm	1. Efficient than the existing algorithms. 2. Improved approach provides high throughput.	1. Running time also increases with the number of pattern increases. 2. Less efficient in space complexity.
3. Wu-Manber algorithm	1. Long random patterns are used. 2. Low matching rate and low memory requirement	1. As worst case complexity is O(n). It takes more number of comparisons. 2. Complex mechanism
4. Boyer-Moore algorithm	1. Suitable for larger strings 2. Simple algorithm is used	1. Worst case complexity is O(n) therefore takes more time to compare. 2. More memory consumption
5. Hash algorithm	1. Takes less execution time because average case complexity O(n/2). 2. needs no preprocessing	1. It has a minimum length problem. 2. Therefore, not suitable for larger strings

VI. PROPOSED METHODOLOGY

There are many multiple pattern matching algorithms but to provide the better results, a comparative study of all of them is necessary. The comparative analysis gives us the clear idea to choose the best algorithm for pattern matching problem. In this paper, the comparative analysis of the most widely used multiple pattern matching algorithms is given. The proposed methodology focuses on the analysis and discussion of multi pattern matching algorithms. There are several multi pattern matching algorithms in use. But it is always cumbersome to select the best one.

For the particularities in pattern matching algorithms and current situation of pattern matching algorithm study, this paper proposes a new type of multi-pattern matching algorithm based on finite state automata, which is combined with frequently matching hash values. The characteristic of the algorithm is using multi-path searching algorithm to match in parallel. One path uses traditional FSA to do multi-pattern matching. The rest paths, try to quickly compare with the frequently matching hash values.

This paper gives the detailed analysis of the most widely used multi pattern matching algorithms. The mostly used algorithms discussed in this paper are: the Aho-Corasick algorithm, the Commentz-Walter algorithm, and the Wu-Manber algorithm. Also, it proposes a new multi pattern matching algorithm which uses frequently used groups. The comprehensive analysis and discussion also helps for the better understanding of the pattern matching problem. Pattern matching is a basic problem in computer science.

The proposed algorithm combines hash technology and multi-pattern matching, and proposes a new type of multi-pattern matching algorithm with automata based on frequently matching hash values. Due to combining the advantages of fast calculation of hash function and parallel pattern matching of automata, it has significant performance advantages in the circumstances of virus signature matching.

Also, in some practical applications where a certain rate of false negative is accepted, we can design these automata to be simpler. The following nodes after the specific node (such as node’d’, 'i+d', 'j+d' in Figure 2) can be removed from the automata. In this way, we can get both time and space performance advantages. This can decrease the possibility of state space explosion of original AC algorithm. The rate of false negative depends on the sum of all entries probabilities in Frequently Used Groups (FUG).

According to the pattern set, construct a FSA similar to AC's, where each node store the pointer (ptr_goto) to Goto, the pointer (ptr_failure) to Failure, and a pointer (ptr_freq) to frequently used group (call FUG for short). After applying pattern matching, a new automata based on Frequently Used Groups is shown in fig. 2.

The algorithm works as shown below:

New multi pattern algorithm based on Frequently Used Groups
Step 1: Initialize Step 2: Construction of Frequently Used Groups (FUG) a) Get the most likely matched patterns when searching to a specific node of the automata. Step 3: Calculate the hash values of these pattern Step 4: store it in the FUG to which this node point to. Step 5: Set the length of hash value be len_hash. Step 6: Set an integer value int_depth, which represents for the depth of current automata node. Step 7: Insert the corresponding FUG to the node whose depth is equal to int_depth. Step 8: Finally, we get a new automata. Step 8: Finish

In step 1, initialize the values to the already constructed finite state automata. On basis of that different pointers are set to different nodes. In step 2, frequently used groups are constructed. For the construction, the most likely

 ISBN: 978-93-86171-47-4

matched patterns are collected when searching to a specific node of the automata. In step 3, hash values of all the patterns are calculated by applying hash function or a message digest. In step 4, hash values are stored into FUG to which the node points to. In step 5, the length of hash value is set to len_hash.

In step 6 an integer value int_depth, which represents for the depth of current automata node is set to current depth. Step 7 inserts the corresponding FUG to the node whose depth is equal to int_depth. Finally, we get a new automaton as shown in fig. 2.

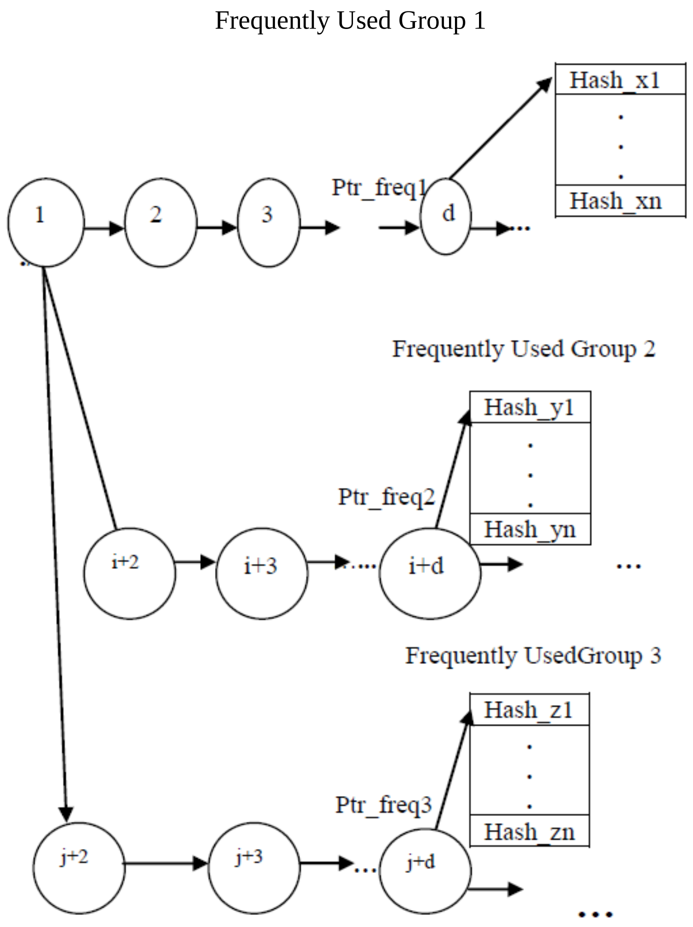

Fig. 2 New automata based on Frequently Used Groups

VII. POSSIBLE OUTCOMES AND RESULT

Results of the proposed algorithm can be shown in two cases:

A. Worst Case

In the worst case, we can't find any matched pattern in the input text sequence, so that the proposed algorithm is equivalent to the original AC algorithm. It needs to compare n nodes of the automata in order, and the time complexity is O(n), where n is the total length of the text sequence.

B. Average Case

The average case, a matched pattern can be found in the input text sequence. As the analysis above, the time complexity of original AC algorithm is O(n). Even if we improve the original one to adapt them to circumstance of virus signature matching, end the algorithm after a matched pattern is found. The time complexity is only up to O(m), where m is the length of the signature.

On the basis of comparative analysis, following results can be drawn. In Aho-Corasick algorithm, if the number of pattern is increases, the running time is also increase. The running time of Commentz-Walter algorithm is also increases with the number of pattern increases. In Wu-Manber algorithm, if the number of pattern is increases, the running time is also increase. But the performance of this algorithm is better than the Aho-Corasick algorithm.

VIII. CONCLUSION

This paper combines hash technology and multi-pattern matching, and proposes a new type of multi-pattern matching algorithm with automata based on frequently matching hash values. Due to combining the advantages of fast calculation of hash function and parallel pattern matching of automata, it has significant performance advantages in the circumstance of virus signature matching. The Aho-Corasick algorithm considers as a classic solution. On the other side, Commentz-Walter algorithm seems to be the first sub-linear running time algorithm for multiple-pattern matching problems in average case by using a sifting technique where a large portion of the text is skipped while searching. The Wu-Manber algorithm has excellent average case performance because of the successful use of shifting operation as a block of characters.

IX. FUTURE SCOPE

In the future improvement, the theory of machine learning can be added to make the FUG be dynamically changing. That is, through long-term learning of the history of matching, dynamically change the entries in FUG to adapt to complex applications. This can further improve the probability of successfully matching in the FUG, so as to improve the overall performance of the algorithm.

REFERENCES

[1] Akinul Islam Jony, **“**Analysis of Multiple String Pattern Matching Algorithms**”**International Journal of Advanced Computer Science and Information Technology (IJACSIT), Vol. 3, No. 4, ISSN: 2296-1739, pp. 344-353, August 2014.

[2] Yu He, Weitong Huang, “IAC: A Real-time Multi-pattern Matching Algorithm Based on Aho-Corsick Algorithm”, 2013 International Conference on Computational and Information Sciences, DOI 10.1109/ICCIS.2013.203, pp. 754-757

[3] Dr. S.P Rajagopalan Dr.V.Anuradha, S. Nirmala Devi, “Index Based Multiple Pattern Matching Algorithm Using Frequent Character Count in Patterns”, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 5, ISSN: 2277 128X, pp. 140-145, May 2013.

[4] Raju Bhukya, DVLN Somayajulu,” Index Multiple Pattern Matching Algorithm using DNA Sequence and Pattern Count”, IEEE Transaction on Information Technology and Knowledge Management, Volume 4, No. 2, pp. 431-441, July-December 2011

[5] S.Nirmala Devi, Dr.S.P Rajagopalan „An Index Based Pattern Matching using Multithreading‟ , International Journal of Computer Applications (0975 – 8887) Volume 50 – No.6, July 2012