Efficient Two Way Backward Non Deterministic Matching (ETNDM) Algorithm

***Hiresh Singh Sengar**

****Prof. B. L. Rai**

*****Prof. Deepak Jain**

ABSTRACT

Bit parallel string matching algorithms are the latest and the efficient class of algorithm for string matching. It uses the intrinsic parallelism inside a computer word is known as bit parallelism. TNDM is the most popular bit parallel string matching algorithm. This algorithm can directly use for various real world problem like as Detection of Intrusion in the network, Data extraction and mine, avoiding Plagiarism, use of Bioinformatics in various medical field. TNDM algorithm having their standard implementation where lots of bit wise operations are checking as not in bit parallel mode like as MSB of bit vector is one or not or the value of the bit vector is zero or non-zero. All these operations are the comparison based operation which are quite time consuming. If these operations are optimized or simplified in bit parallelism based operations than they have their best bit parallelism based implementation.

In this paper we are found out such operations and simplified in pure bit parallel based implementation. Here we simplified the popular algorithm TNDM and named as Efficient Two Way Backward Non Deterministic Matching (ETNDM) Algorithm. The working of Efficient Two Way Backward Non Deterministic Matching (ETNDM) Algorithm is same as classical one but due to use of bit wise operation in place of comparison operator our algorithm gives better result. Experimental results show that our ETNDM algorithm is better than the standard implementation.

Keywords: String Matching, Bit Parallelism, TNDM, Efficient TNDM.

*Hiresh Singh Sengar, M. Tech Scholar, Computer Science Department, JNCT Bhopal, (M.P.), hireshtech@gmail.com

**Prof. B. L. Rai, (Guide) Dean Administration, JNCT Bhopal, (M.P.), blrai_08_76@yahoo.com

***Prof. Deepak Jain, HOD, Computer Science Department, JNCT Bhopal, (M.P.), deepak.jain25@gmail.com

INTRODUCTION:

String Matching Algorithms [1] are used in various real world applications whether it is used directly or indirectly. So improve in the performance of these string matching algorithms the efficiency of these real world application like as Intrusion Detection System [2, 3], Data Mining [4, 5], Digital Forensics [6, 7], Plagiarism Detection [8, 9], Bio Informative [10] Video Retrieval[11] is also improved. Before nineties there are various character based algorithms that

set their benchmarks in string matching and used like as Boyer Moore [12], BMH [13], Rabin Karp [14], Wu Manber[15] and Aho-Corasik[16]. By using the character based algorithms the efficiency of these algorithm can be improved up to certain level so improve in efficiency of sting matching algorithm Bit parallel string matching algorithm [17] are used which uses the intrinsic parallelism of the computer. There are various bit parallel string matching algorithms are used like as BNDM [18], TNDM [19], BNDMq[20], Shift OR [21], Shift OR with q gram [22]. Here, we present an Efficient TNDM. In our Efficient TNDM algorithm we try to improve the loop of algorithm as well as use as much as bitwise operations like checking the bit vector is zero or not and the value of MSB is one or zero. These operations in classical algorithm are carried out with the help of comparison operators here, we use bitwise operator instead of comparison operators. Our experiments have shown that Efficient TNDM algorithm is much faster than compared to original one for the large text data.

IMPROVEMENT PARAMETER

Character Mismatch

Character mismatch means pattern is not found at that position. It is important to identify to shift the window. In character based algorithms it can be checked directly but in case of bit parallel algorithms, it is checked by the value of bit vector. It means if value of bit vector D become zero. This is checked by simply comparison operator which is time consuming. So, instead of comparison operator we can use bit wise operator which can improve the performance.

Pattern Found

Similar to pattern mismatch pattern match is important which is clearly identifiable in character based algorithm while in bit parallel it is identified by the checking the MSB of the bit vector. In the case of TNDM algorithm and other bit parallel algorithm pattern are found when MSB is one. This checking of MSB is carried out by the comparison operator but in efficient TNDM algorithm we use the bit wise operator to get better results.

Iteration Minimization

In most of the bit wise algorithm like BNDM, TNDM, BNDMq we create loops to find the patterns in the text. These loops take lot of time to execute because it runs many times according to the condition until it is false. By doing some statement changes or conditions there is a possibility of getting efficient results. Because in the TNDM algorithm the value of D is left shift

and compared with the limit to MSB check after the value of D become zero means no possibility to find the pattern. In the efficient TNDM algorithm these extra work is cut out from the algorithm

EFFICIENT TNDM ALGORITHM

TNDM [19] is a variation of the BNDM algorithm. With respect to the working of TNDM algorithm, the whole concept is same as BNDM algorithm instead of pattern mismatch at first step. So, we implement the efficient TNDM algorithm in which when mismatch occurs at very first position it search suffix same as the classical TNDM one for prefix of the pattern. But in remaining case it searches as efficient TNDM algorithm where we use bitwise operator instead of comparison operation. There is another change in the loop through which the number of comparisons decreases up to a certain level and we get better results. Table 1 shows the parameter changes between classical TNDM and Efficient TNDM algorithm.

TABLE I

Change Parameter of TNDM Algorithm

Change Parameter	Classical TNDM	Improved TNDM
Mismatch (D all Zero)	D != 0	! D
Pattern Found (MSB Checking)	D >= Limit	D & 10^{m-1}
Iteration Minimization	After D is Zero It Check MSB Condition as well as Shift left D	After D is Zero it will come out of loop

Efficient TNDM algorithm after applying the basic parameter changes are shown below. Where p is the pattern of length m and T is the text of length n

IMPROVED TNDM ALGORITHM

IMPROVED TNDM(P:p1-pm, T:t1-tn)

Preprocessing

1. for c ∈ ∑ do B[c] ← 0^m
2. for j ∈ 1 . . .m do
3. B[pj] ← B[pj] | $0^{j-1}10^{m-j}$
4. Initshift (P, restore [ ])

Searching

5. epos ← m
6. while epos ≤ n do
7. i ← 0; last ← m
8. D ← B[t_{epos}]

```
9.    if (!D) then /* when D  ≠ B[pm], */
10.        do /* forward scan */
11.              i ← i + 1
12.              D ← D&(B[tepos+i] << i)
13.           until ( ! D  and D&10^i )
14.       if D = 0^m then // already last ← m
15.              goto Over
16.        epos ← epos + i; last ← restore[i]
17.          while j > 0 do
18.            D ←D&B[T_pos+j]
19.            if (!D) goto Over
20.            j--;
21.            if (D & 10^m-1 )
22.                last← j
23.           D ←D<< 1
24.         end while
25.         report an occurrence at pos + 1
26.         Over:
27.            epos←epos + last
//Initializations of the array restores.

Initshift (P = p1p2 · · · pm, restore[ ])

1. D ← 1^m
2.   last ← m
3.   for i ← m down to 1 do
4.      D ← D&B[pi]
5.      if ( D&10^m−1 ) then
6.          if i > 0 then last ← i
7.     restore[m − i + 1] ← last
8.     D ← D << 1
```

TIME AND SPACE COMPLEXITY

Time complexity of the algorithm can be calculated by these different scenarios. Let 'T' be the text of length n and 'P' is the pattern of length m so we have to find P in T. Let us assume that $m \leq w$ holds, then the preprocessing time complexity is $O(m+|\Sigma|)$,where $|\Sigma|$ is number of distinct characters existing in pattern. The worst case when all the text as well as pattern are made of single character(when T=an p=am) time complexity of Efficient TNDM is same as that of original one i.e. O(mn). In an average case scenario, the inner loop of this Efficient TNDM algorithm has a frequency lesser than the frequency count of classical TNDM. Thus, on an average case this Efficient TNDM performs far better than classical TNDM. The best case (when T=an P=am-1b) time complexity of this Efficient TNDM is O(n/m) which is same as TNDM. However, our algorithm benefit by reducing the number of comparison's up to 30% in best case. Efficient TNDM need occurrence vectors B for each character which is same as classical TNDM. So we can say that the space complexity is unchanged in efficient TNDM algorithm.

EXPERIMENTAL RESULTS

This section gives the brief introduction to the experimental results and analysis. In this a long text and patterns of different length are taken and running time of algorithms improved algorithms have been calculated and on the basis of these results a table of comparison is formed as well as graphs which clearly represents that the Efficient TNDM algorithm has got considerable speedup as compared to classical TNDM and other variant of bit parallel as well as character based algorithm.

Experimental Environment

All the algorithms are tested on the system having following configuration:

Processor: 2.1 GHz Intel I3 CPU, 2.10 Ghz; RAM: 3 GB; L1 Cache: 128 KB; L2 Cache: 512 kB; L3 Cache: 3 MB; System Type: 32 Bit Operating System OS: windows 7

Implementations are done using Visual studio 2010 platform and developed in Visual C++.

Experiment Data

All the implementations are tested on text file of size 210 MB having large number of occurrences of pattern and five different Patterns of length 4, 8, 12, 16 and 32 bit. In the text file are taken from bible [23] and patterns are randomly taken from bible itself.

Efficient TNDM Algorithm

Efficient TNDM algorithm is compared with classical TNDM algorithm for different pattern sizes as shown in Table 2. And comparison of execution time of Efficient TNDM algorithm with classical TNDM is shown in Figure 2.

TABLE II

Comparison between TNDM & Efficient TNDM

Pattern Size	TNDM	Efficient TNDM
	Execution Time (mSec)	
4 bit	598	460
8 bit	513	397
12 bit	405	323
16 bit	273	214
32 bit	165	105

These table and figure are clearly shows that the our algorithm is better than the classical one and take lesser time as compare to existing one.

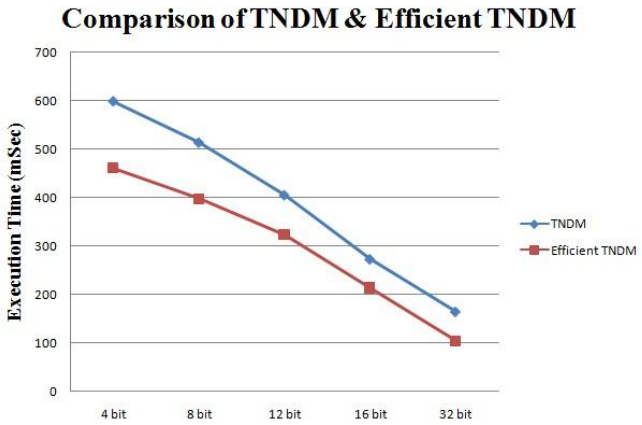

Fig. 1 Comparative Analysis of TNDM and Efficient TNDM

Table 3 shows the comparison among the efficient TNDM algorithm to the other bit parallel algorithm on the 210 MB data and we get the improved results which is shown in the figure 2 with the help of graph chart.

TABLE III

Comparison between various bit parallel and ETNDM algorithm

Pattern Size	Efficient TNDM	SBNDM	FBNDM
	Execution Time (mSec)		
4	460	601	593
8	397	508	467
12	323	426	411
16	214	293	267
32	105	167	149

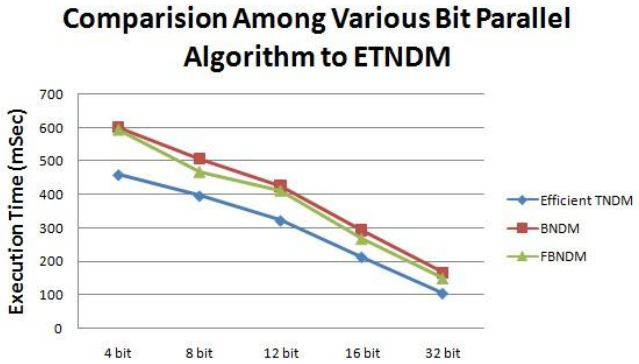

Fig. 2 Comparison among various bit parallel algorithm to ETNDM

Table 4 shows the comparison among various character based algorithm and Efficient TNDM algorithm and data obtained is clearly shows that the our algorithm is far better than the character based algorithm which is also shown in figure 3 with the help of line graph.

TABLE IV

Comparison between character based and ETNDM algorithm

Pattern Size	Efficient TNDM	Boyer Moore	Aho Corasick
	Execution Time (mSec)		
4	460	703	697
8	397	636	574
12	323	561	478
16	214	408	376
32	105	309	305

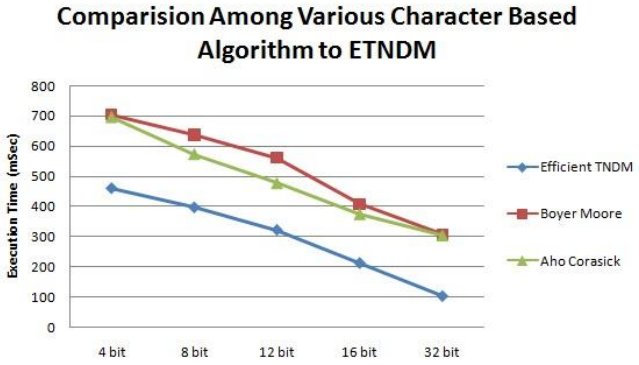

Fig. 3 Comparison among various character based algorithm to ETNDM

CONCLUSION

We have proposed and developed Efficient Two Way Backward Non Deterministic Matching (ETNDM) Algorithm to improve the efficiency of the Two Way Backward Non Deterministic Matching (TNDM) Algorithm. In this paper we are found out such operations which can be improved like as comparison operator takes time so these operators improved in pure bit parallel based implementation. Here we improved the popular algorithm TNDM. The working of improved version of this algorithm is same as classical one but due to use of bit wise operation in place of comparison operator our algorithm gives better result. Experimental results and analysis shows that our simplified implementation is the better than the standard implementation.

FUTURE WORK

We have performed detailed study on several bit parallel string matching algorithms. Bit parallel string matching algorithms are the faster algorithms among all string matching algorithms. Mostly bit parallel algorithms are single pattern matching algorithms so it can be implemented in multiple pattern matching algorithms. We can also effectively remove the limitation of word size present in the most of the bit parallel algorithms. These algorithms use bit wise operation so it can be easily implemented in the hardware. For further improvement on GPGPUs memory optimization can be done.

REFERENCES

1. Christian Charras and Thierry Lecroq, "Handbook of Exact String Matching Algorithms", Published in King's college publication, Feb 2004.
2. Hyunjin Kim, Hong-Sik Kim and Sungho Kang, "A Memory-Efficient Bit-Split Parallel String Matching Using Pattern Dividing for Intrusion Detection Systems" IEEE Transactions on Parallel and Distributed Systems, Volume:22 , Issue: 11, pp. 1904-1911, Nov 2011.
3. Pei-fei Wu and Hai-juan Shen, "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.
4. 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.

5. Qiong Zhang, Roger D. Chamberlain, Ronald S. Indeck, Benjamin M. West and Jason White, "Massively Parallel Data Mining Using Reconfigurable Hardware: Approximate String Matching", In Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS'04), 2004.
6. Jooyoung Lee, Sungkyung Un, and Dowon Hong, "Improving Performance in Digital Forensics", In the Proc. of International Conference on Availability, Reliability and Security, 2009.
7. Nicole Lang Beebe, Jan Guynes Clark, "Digital forensic text string searching: Improving information retrieval effectiveness by thematically clustering search results", digital investigation 4S S49 – S54, 2007.
8. 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.
9. Alzahrani, S.M, Saudi Arabia, Salim, N and Abraham, A, "Understanding Plagiarism Linguistic Patterns, Textual Features, and Detection Methods", IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, Volume:42 , Issue: 2, March 2012.
10. Lok-Lam Cheng, David W. Cheung and Siu-Ming Yiu, "Approximate String Matching in DNA Sequences", In Proceedings of the Eighth International Conference on Database Systems for Advanced Applications (DASFAA'03), 2003.
11. JunWei Hsieh, Shang-Li Yu and Yung-Sheng Chen. "Trajectory-based Video Retrieval by String Matching", In the Proc. of International Conference on Image Processing (UP), 2004.
12. BOYER, R. S. AND MOORE, J. S, "A fast string searching algorithm", Communication of ACM 20, Vol. 10, pp. 762–772, 1977.
13. HORSPOOL, R. N, "Practical fast searching in strings", In proc. Of Software Practical Exp, Vol. 10, 6, pp. 501–506, 1980.
14. Richard M Karp and Michael O Rabin, "Efficient randomized pattern-matching algorithms", IBM Journal of Research and Development, vol. 31(2), pp. 249-260, March 1987.
15. 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.
16. Alfred v. aho and margaret j. corasick, "efficient string matching: an aid to bibliographic search" communication of acm, vol. 18, june 1975.

17. 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.
18. G. Navarro, M. Raffinot, "Fast and flexible string matching by combining bit-parallelism and suffix automata",ACM Journal. Experimental Algorithmics 2000.
19. Hannu Peltola and Jorma Tarhio, Alternative Algorithms for Bit-Parallel String Matching, String Processing and Information Retrieval, Spire 2003Springer, LNCS 2857,pp. 80-93, 2003.
20. 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
21. R. Baeza-Yates and G. Gonnet, "A new approach to text searching", Communication of ACM, Vol. 35(10), pp. 74–82, 1992.
22. L. Salmela, J. Tarhio, and J. Kytojoki, "Multi pattern string matching with q-grams", Journal of Experimental Algorithms, Volume 11, pp. 1-19, 2006.
23. www.holybooks.com/download-bible.