American J. of Engineering and Applied Sciences 4 (1): 102-107, 2011
ISSN 1941-7020

A Fast Hybrid Algorithm for the Exact String Matching Problem

Abdulwahab Ali Al-mazroi and Nur'Aini Abdul Rashid
Parallel and Distributed Computing Center, School of Computer Sciences,
University Science Malaysia, 11800, Penang, Malaysia

Abstract: Problem statement: Due to huge amount and complicated nature of data being generated recently, the usage of one algorithm for string searching was not sufficient to ensure faster search and matching of patterns. So there is the urgent need to integrate two or more algorithms to form a hybrid algorithm (called BRSS) to ensure speedy results. **Approach:** This study proposes the combination of two algorithms namely Berry-Ravindran and Skip Search Algorithms to form a hybrid algorithm in order to boost search performance. **Results:** The proposed hybrid algorithm contributes to better results by reducing the number of attempts, number of character comparisons and searching time. The performance of the hybrid was tested using different types of data-DNA, Protein and English text. The percentage of the improvements of the hybrid algorithm compared to Berry-Ravindran in DNA, Protein and English text are 50%, 43% and 44% respectively. The percentage of the improvements over Skip Search algorithm in DNA, Protein and English text are 20%, 30% and 18% respectively. The criteria applied for evaluation are number of attempts, number of character comparisons and searching time. **Conclusion:** The study shows how the integration of two algorithms gives better results than the original algorithms even the same data size and pattern lengths are applied as test evaluation on each of the algorithms.

Key words: Hybrid algorithm, string matching, pre-processing phase, searching phase

INTRODUCTION

String matching algorithm is an essential segment in computer science presently because of its usefulness in searching and matching pattern and text from vast databases containing huge of complicated data (Hassan, 2005). String matching algorithm effectiveness is spread over a wide range of problem solving areas such as computer applications, text processing, artificial intelligence, information security and among others (Raju and Babu, 2007). Due to increasing rate and complex nature of biological sciences and scientific data nowadays the usage of one algorithm alone for string searching is not efficient thus the urgent need to combine two or more algorithms to form a hybrid in order to ensure efficient performance (Chen, 2007). These concerns have arose the interest of researchers who are coming up with new hybrid algorithms as the application of one algorithm for string searching cannot meet today's complicated and large of data being generated currently (Michailidis and Margaritis, 2002).

This study therefore proposes a new hybrid algorithm (called BRSS), by merging the best properties of two algorithms specifically Berry-Ranvindran and Skip Search algorithms in order to ensure a better performance during string searching. These algorithms are chosen because Skip Search algorithm is more efficient for small alphabets and long patterns, while Berry-Ravindran algorithm is more effective for providing a better shift value from the two successive characters immediately to the rightmost of the window (Tathoo *et al*., 2006). An exact pattern matching involves the identification of all occurrences of a given pattern of m characters (X = x1, x2…xm) built over a finite alphabet Σ size of σ.

The principal concern of these algorithms is on how to slide the window to match the pattern within a given large pool of texts during the searching phase. The algorithm is basically operated by scanning a given set of texts through the assistance of a window, where the window has the length of the pattern when a match or mismatch occurs. After the texts in the window are arranged in lines, the characters in the text are compared with a pattern until a match is found. During the searching phase, there would be either a match or mismatch as is normally the case, such that the window is shifted to the rightmost section for the alignment of text which begins from left to right at the beginning of the search (Mohammad *et al*., 2006).

Corresponding Author: Abdulwahab Ali Al-mazroi, School of Computer Sciences, University Science Malaysia, 11800 Penang, Malaysia Tel: +604-6533640 Fax: +604-6573335

Related study: Previous study undertaken by researchers show that combining the best features of two good algorithms to form a hybrid algorithm help immensely in increasing the performance levels during searching phase (Madan and Madan, 2010), because they both complement each other weaknesses.

Franek *et al*. (2005) is hybrid algorithm which comprised of Boyer-Moore and Knuth-Moris-Pratt Algorithms. The simple method of the algorithm is that; the pattern is firstly lined up against the text in order that m[1] is in line with y[1]. For every line up of the pattern, the pattern is tested 2 against the text to find out if there is a match occurring at that specific area. The shifting of the pattern is done at one or more positions to the end of texts which are aligned. These procedures are repeat till there is no possible match, by that time p[m] "slides off the end" of the text. Additionally, BRFS (Huang *et al*., 2008) algorithm is also hybrid algorithm made up of Fast Search and Berry-Ravindran algorithms. This algorithm applies two operations for the pre-processing stage namely Boyer Moore good suffix function represented by bmGs(i) and Berry-Ravindran bad character represented by brBc(x,y), which is the computation of the two successive characters after the current window. The searching phase applied the Fast Search algorithm which compares the pattern and text starting from left to right till either a match or a mismatch occurs.

BRQS (Klaib and Osborne, 2009) hybrid algorithm includes two combined algorithms that are Berry-Ravindran and Quick Search algorithms. The shift value depends on the two successive characters immediately after the window for shifting the pattern during the pre-processing phase. In the searching phase, comparison is performed at the rightmost character in the pattern with another rightmost text of the window. The function of the search moves back the pattern one character at a time till leftmost character is reached for a pattern and a text of the window.

MATERIALS AND METHODS

This research started by analyzing all the major algorithms, before deciding on Berry-Ravindran and Skip Search algorithms, by extracting their best properties from both algorithms namely bad character table for Berry-Ravindran and bucket for Skip Search. The computation of the shift value in the pre-processing phase is to ensure a bigger shift value for the window during the searching phase of the hybrid algorithm. By joining the two algorithms, each one will complement the other's weaknesses. Thereby, this will increase the performance of the new algorithm. Berry-Ravindran provides the best shift value using two successive characters. However, the disadvantage of this algorithm is that it does not check the starting point as a first step. In the case of Skip Search, the strength of this algorithm is that it begins by checking the beginning position in the text before starting the searching phase while the weakness of this algorithm is that it uses all the positions of the examining characters in case of match or mismatch.

Hybrid algorithm pre-processing phase: The hybrid algorithm build the pre-processing phase first (Zalata and Alqadi, 2007) and at this phase the hybrid algorithm consists of building two tables namely bad character table and bucket list by computing the shift values. The first table to be used in the pre-processing phase of hybrid algorithm is bad character table; this bad character table is built by using Berry-Ravindran formula as shown in Eq. 1:

$$\mathrm{brBc}[m,n] = \min\left\{\begin{array}{ll} 1 & \text{If } x[m-1]=m \\ m-i+1 & \text{If } x[i]\times[i+1]=mn \\ m+1 & \text{If } x[0]=n \\ m+2 & \text{Otherwise} \end{array}\right\} \quad (1)$$

And the second table to be created is bucket list, which contains all the locations of the characters that exist in the pattern and the text.

Hybrid algorithm searching phase: The searching phase of the hybrid algorithm begins by applying the pattern length to start the operation of the searching phase. The processes are described as follow:

- Scan the m-length character of the text to demarcate a possible beginning search point
- If the examining character is not exist in the bucket, first calculate the bad character shift value by using the rightmost two consecutive characters after the examining m-length character
- If the bad character shift value ≥ Pattern length, use the bad character shift value, otherwise, apply the shift value of pattern length
- If the scanning character exists in the bucket, arrange the character of initial search point and the pattern with the equivalent location of the character in the bucket
- Begin the comparing of characters from left to right
- When there is a match or mismatch happens, compute the shift value of the Skip Search in the first instance and secondly, calculate the bad character shift value from the two rightmost consecutive characters immediately after the window

- If the bucket shift value ≥ bad character shift value, use the bucket shift value, otherwise, use the bad character shift value
- If the corresponding character is in the last position in the bucket, first calculate the bad character shift value from the rightmost two consecutive characters immediately after the window
- If the bad character shift value ≥ Pattern length, use the bad character shift value, otherwise, apply the shift value of the pattern length

Analysis: The hybrid algorithm contains the pre-processing phase of Skip Search and Berry-Ravindran algorithms. Therefore, the pre-processing phase time complexity of the hybrid algorithm is O $(m+\sigma^2)$. The searching phase time complexity is categorized as follows.

Lemma 1: The time complexity is O(mn) in the worst case.

Proof: The worst case in the hybrid algorithm implies that all characters within text are matched, which should not be greater than m times. This worst case normally happens during the process when all the characters within the pattern are the same to the other characters in the text. For example given text T = "dddddddddddddd" and pattern P =”ddddd" and so consequently the time complexity is O(mn) in the worst case.

Lemma 2: The time complexity is O(n/(m +2)) in the best case.

Proof: in every attempt, when the examine character did not exist in the Skip bucket, the shift value will be m+2 as stated through Berry Ravindran function calculated during the pre-processing phase. In the best case, when every single character within the pattern are uniquely different from the characters in text. For example given text T = "ccccccccccccccccccc" and pattern P = ”yyyyyyy", there is the shift which is equal to m+2, performed at every attempt throughout the searching phase, so accordingly the time complexity is O(n/(m +2)) in the best case.

The basic elements which determine the average time complexity are the size of the alphabet and the possible occurrence of every single character in the text. Accordingly, the maximum shift that can be accomplished is m+2 and the minimum would be one; character comparisons could be between 1-m, which is totally random based on the input data. In view of this random nature with no dependable estimation tool, the average time complexity is impossible to predict in this case.

Evaluation: The hybrid and the original algorithms are tested on three different kinds of data in order to appraise the random disparities for every algorithm and the average value is chosen for each algorithm. These data are chosen due to their standardization benchmarks (Mahmood *et al*., 2009) and the usefulness in testing algorithms and their related behaviors when giving different sizes of alphabets and patterns. Accordingly, the program is executed 6 times. The data used in the evaluation is DNA data contain alphabet size equal to four letters ($\sigma = 4$), Protein data contain alphabet size equal to twenty letters ($\sigma = 20$) and English text made up of 100 kinds of alphabets representing all the English language characters, numbers and symbols. The DNA data and English text are acquired from Gutenberg project, while Protein data is acquired from Swiss-Prot Database. The algorithms implemented on Personal Computer with operating system being Microsoft Windows Vista Service Pack 2 with 1.93 GHz Intel Core 2 Duo Processor, 3GB of RAM and programming editor C++ Builder 2010 Architect. The hybrid, Skip Search and Berry-Ravindran algorithms are evaluated in terms of number of attempts, number of character comparisons and searching time.

Number of attempts: Is the starting point where the pattern of first character is mapped to a particular character within the text and continues to shift till the end of the text, so as to determine whether a match or mismatch occurs before the text ends. The estimated number of shifting to the end of the text is the number of attempts (Hudaib *et al*., 2008).

Number of character comparisons: This is the beginning point within a given text to the last letter of the text, where the pattern characters are extracted individually and compared with the text characters to determine whether there is a match or mismatch (Tathoo *et al*., 2006).

Searching time: the time taken in searching the whole length of a given text to find out at the end of the search whether there is a match or a mismatch of characters within a given text (Kalsi *et al*., 2008).

RESULTS

The hybrid and the original algorithms are tested using data sizes of 200MB. Moreover the performance

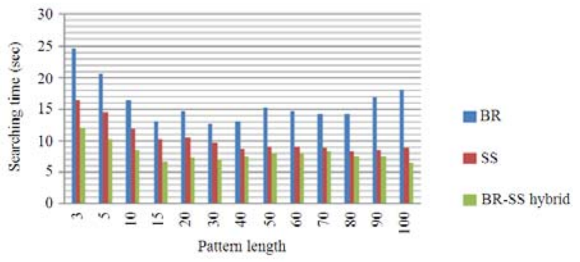

Fig. 1: Searching time in DNA data

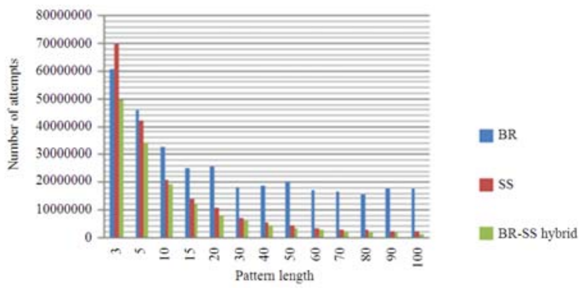

Fig. 2: Number of attempts in DNA data

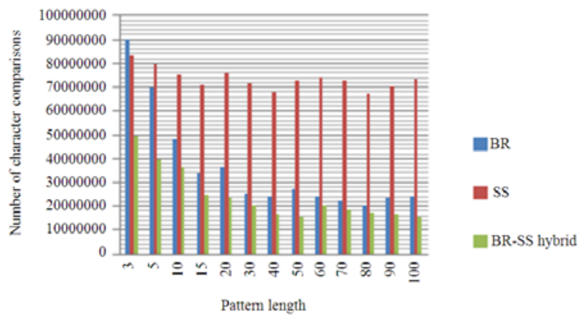

Fig. 3: Number of character comparisons in DNA data

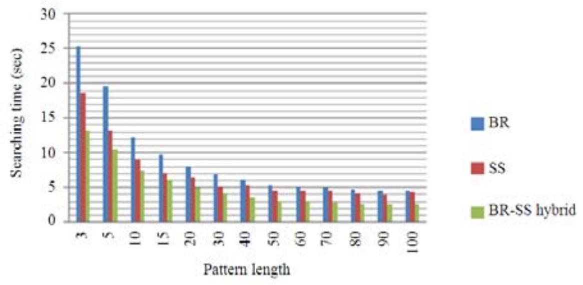

Fig. 4: Searching time in protein data

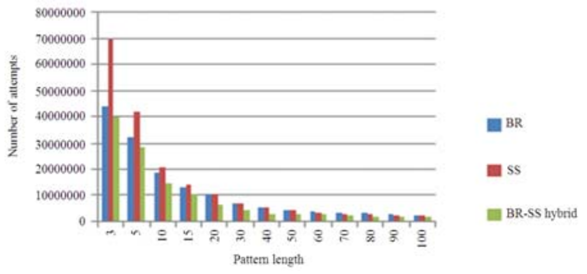

Fig. 5: Number of attempts in protein data

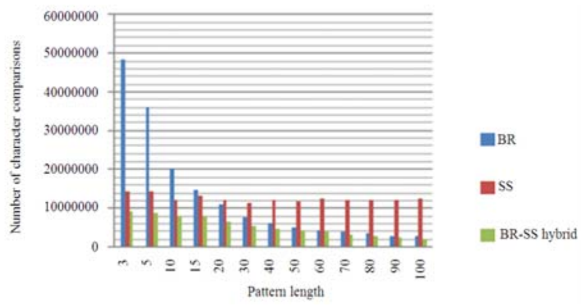

Fig. 6: Number of character comparisons in protein data

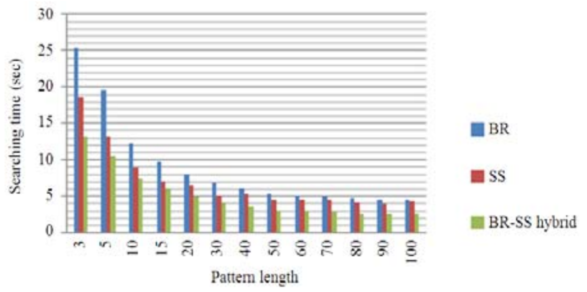

Fig. 7: Searching time in English text

of the algorithms is evaluated by applying different pattern lengths; these are 3, 5, 10, 15, 20, 30, 40, 50, 60, 70, 80, 90 and 100 characters. From Fig. 1-9 illustrate the three criteria of evaluation namely number of character comparisons, number of attempts and searching time.

The results from Fig. 1-9 imply that the hybrid algorithm shows reduced levels of number of character comparisons, number of attempts and searching time, thus the combination of the two algorithms have made good use of the best features extracted to form the hybrid algorithm as they are improved performances. For instance, the results for the DNA, Protein and English text of the hybrid algorithm displayed enhanced performance than the original algorithms; it is largely due to combination of the two tables of Berry-Ravindran bad character and Skip Search bucket, enables a bigger shift values for the window thereby increasing the performances for the hybrid algorithm.

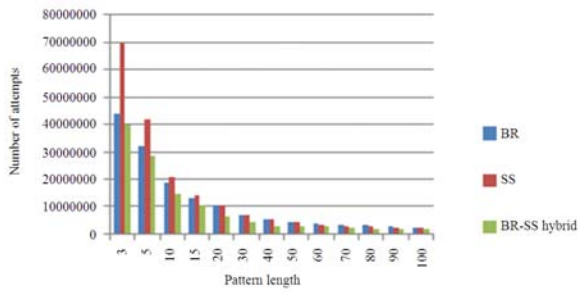

Fig. 8: Number of attempts in English text

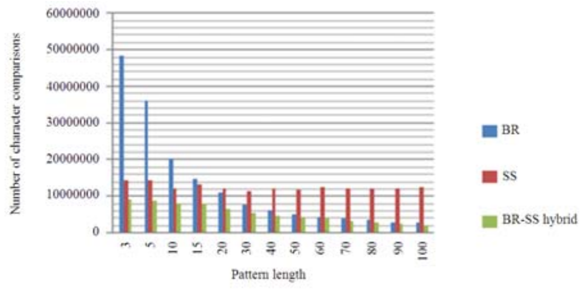

Fig. 9: Number of character comparisons in English text

DISCUSSION

As explained earlier the objectives were to integrate two different algorithms to form a hybrid algorithm so as to increase the performance during pre-processing and searching phases as scientific data are very intricate and voluminous. So to gauge the performances of the new hybrid algorithm three different set of data were applied namely DNA, Protein and English text. And the results demonstrated how with Berry Ravindran-Skip Search hybrid algorithm reduced the number of attempts, number of character comparisons and searching time and as well as increased the performances. And the performances of the new hybrid algorithm improved between 50%, 43% and 44% over Berry-Ravindran Algorithm and at the same time also improve from 20%, 30% and 18% over Skip Search Algorithm respectively when the three different data were applied.

The outcome implies that hybridization of the two algorithms is the way forward for the future as the new hybrid algorithm led to faster processing times and diminishing number of attempts, comparisons and searching times than the use of one algorithm for which is ineffective for searching gigantic data as technology develops and data becomes complex. The success for the new hybrid algorithm also indicates, choosing the right algorithms are of prime of importance when deciding which algorithm(s) to select to form hybrid as some of them cannot perform creditably even when combine together. The results also established the applicability of merging two search algorithms to form hybrid algorithm and the benefits derived through enhance searching performances.

From this we strongly recommend the application of hybrid algorithms for future development or even to a larger extent the combination of two hybrid algorithms to further boost performances to higher levels.

CONCLUSION

This study presents a new hybrid algorithm called BRSS, by combining two algorithms, Berry-Ravindran and Skip Search. The hybrid algorithm demonstrates enhanced character comparisons, number of attempts and searching time performances in all the different data size and pattern lengths, therefore the proposed algorithm is useful for searching DNA, Protein and English text. This also proved that the application of the hybrid algorithm will lead to better searching and matching of the patterns than the use of one algorithm as data is becoming more complex presently.

REFERENCES

Chen, Y., 2007. A new algorithm for subset matching problem. J. Comput. Sci., 3: 924-933. DOI: 10.3844/jcssp.2007.924.933

Franek, F., C.G. Jennings and W.F. Smyth, 2005. A simple fast hybrid pattern-matching algorithm. Combinatorial Patt. Match., 3537: 13-70. DOI: 10.1007/11496656_25

Hassan, A.A., 2005. Mixed heuristic algorithm for intelligent string matching for information retrieval. Proceedings of the 6th International Conference on Computational Intelligence and Multimedia Applications, Aug. 16-18, Reading University, UK., pp: 11-16. DOI: 10.1109/ICCIMA.2005.39

Huang, Y., L. Ping, X. Pan and G. Cai, 2008. A fast exact pattern matching algorithm for biological sequences. Proceedings of the International Conference on Biomedical Engineering and Informatics, May 27-30, Sanya, pp: 8-12. DOI: 10.1109/BMEI.2008.154

Hudaib, A., R. Al-Khalid, D. Suleiman, M. Itriq and A. Al-Anani, 2008. A fast pattern matching algorithm with Two Sliding Windows (TSW). J. Comput. Sci., 5: 393-401. DOI: 10.3844/jcssp.2008.393.401

Kalsi, P., H. Peltola and J. Tarhio, 2008. Comparison of exact string matching algorithms for biological sequences. Bioinform. Res. Dev., 13: 417-426. DOI: 10.1007/978-3-540-70600-7_31

Klaib, A.F. and H. Osborne, 2009. BRQS matching algorithm for searching protein sequence databases. Proceedings of the International Conference on Future Computer and Communication, Apr. 3-5, Kuala Lumpar, pp: 223-226. DOI: 10.1109/ICFCC.2009.40

Mohammad, A., O. Saleh and R.A. Abdeen, 2006. Occurrences algorithm for string searching based on brute-force algorithm. J. Comput. Sci., 2: 82-85. DOI: 10.3844/jcssp.2006.82.85

Mahmood, A.W., N.A. Rashid and A.A. Rozaq, 2009. BM-KMP hybrid algorithm for exact and subsequence string matching. Proceedings of the 3rd International Conference on Informatics and Technology, USM, Kuala Lumpur Malaysia, Oct. 27-28, pp: 81-87.

Madan, M. and S. Madan, 2010. Convalesce optimization for input allocation problem using hybrid genetic algorithm. J. Comput. Sci., 6: 413-416. DOI: 10.3844/jcssp.2010.413.416

Michailidis, P.D. and K.G. Margaritis, 2002. On-line approximate string searching algorithms: Survey and experimental results. Int. J. Comput. Math., 79: 876-888.

Raju, S.V. and A.V. Babu, 2007. Parallel algorithms for string matching problem on single and two dimensional reconfigurable pipelined bus systems. J. Comput. Sci., 3: 754-759. DOI: 10.3844/jcssp.2007.754.759

Tathoo, R., A. Virmani, S.S. Lakskmi, N. Balakrishnan and K. Sekar, 2006. TVSBS: A fast exact pattern matching algorithm for biological sequences. J. Current Sci., 91: 47-53.

Zalata, M.A. and Z. Alqadi, 2007. Separating low pass and high pass frequencies in the image without loosing information. Am. J. Applied Sci., 4: 237-244. DOI: 10.3844/ajassp.2007.237.244