RESEARCH ARTICLE OPEN ACCESS

A Novel Framework for Short Tandem Repeats (STRs) Using Parallel String Matching

D. Bala MuraliKrishna, Ch. Someswara Rao.
Department of Computer Science, S.R.K.R Engineering College, Bhimavaram, AP, India.
Associate Professor, Department of Computer Science, S.R.K.R Engineering College, Bhimavaram, AP, India.

Abstract
Short tandem repeats (STRs) have become important molecular markers for a broad range of applications, such as genome mapping and characterization, phenotype mapping, marker assisted selection of crop plants and a range of molecular ecology and diversity studies. These repeated DNA sequences are found in both Plants and bacteria. Most of the computer programs that find STRs failed to report its number of occurrences of the repeated pattern, exact position and it is difficult task to obtain accurate results from the larger datasets. So we need high performance computing models to extract certain repeats. One of the solution is STRs using parallel string matching, it gives number of occurrences with corresponding line number and exact location or position of each STR in the genome of any length. In this, we implemented parallel string matching using JAVA Multi-threading with multi core processing, for this we implemented a basic algorithm and made a comparison with previous algorithms like Knuth Morris Pratt, Boyer Moore and Brute force string matching algorithms and from the results our new basic algorithm gives better results than the previous algorithms. We apply this algorithm in parallel string matching using multi-threading concept to reduce the time by running on multicore processors. From the test results it is shown that the multicore processing is a remarkably efficient and powerful compared to lower versions and finally this proposed STR using parallel string matching algorithm is better than the sequential approaches.

Keywords: computing model, DNA, STR, parallel string matching, multicore processing.

I. Introduction

DNA molecules are subject to a variety of mutational events. One of the less well understood is tandem duplicationin which a stretch of DNA, which we call the pattern, is converted into two or more copies, each following the preceding one in a contiguous fashion. For example we could have consider the below String TCGGCGGCGGA, and the pattern CGG.

. . . TCGGA . . . $\longrightarrow$ . . TCGGCGGCGGA . . .

In which the single occurrence of triplet CGG has been transformed into three identical, adjacent copies. The result of a tandem duplication event is termed a tandem repeat. Over time, individual copies within a tandem repeat may undergo additional, uncoordinated mutations so that typically, only approximate tandem copies are present. Tandem repeats are presumed to occur frequently in genomic sequences, comprising perhaps 10% or more of the human genome [1]. But, accurate characterization of the properties of tandem repeats has been limited by the inability to easily detect them. In recent years, the discovery of the trinucleotide repeat diseases has piqued interest in tandem repeats.

STRs (Short Tandem Repeats) are the genetic loci where one or few bases are repeated for varying numbers of times. Such repetitions occur primarily due to slipped-strand impairing and subsequent error(s) during DNA replication, repair, or recombination [1]. STRs comprising 2–6 base pairs long, occur frequently and are ubiquitously interspersed in many genomes [2, 3]. The biological importance of STR tracts has been clearly delineated. STR loci show extensive length polymorphism, and hence they are widely used in DNA fingerprinting and diversity studies [4, 5]. They are also considered as ideal genetic markers for the construction of high-density linkage maps.In spite of its high significance, a bioinformatics tool for the analysis of these regions is not available.

Available algorithms directly or indirectly detect tandem repeats. However, there are many limitations with these algorithms [6].The drawbacks are high computational time required by the algorithm and their inability to predict the positions of STRs in the genome.Finally, researchers have to understand the classical methods of pattern matching to develop new efficient algorithms [7-10]. In this work, we describe a program called STRs using parallel string matching that uses Multi-threading to find STRs on multicore processors. The program Framework for STRs using parallel string matching locates repeats with patternsexact positon, line number and number of occurrences of large number of files and directories taking as input.

A serious comparison between different tools should, of course, be performed by a third party against a sufficient number of carefully selected test problems, carefully tuning many different search parameters; meanwhile we can say that in the case of naive runs with default parameters on easy problems, all the above programs appear to output basically equivalent results. With the developments of new string matching techniques, efficiency and speed are the main factors in deciding among different options available for each application area [11-13].However, a major drawback of all such tools is that, when running against very long sequences, they produce a large amount of cumbersome results that require a painstaking interpretation. In order to provide the user with some capability of grasping the STRs at first glance and looking over the drawbacks for large sequencing files, we decided that the only way to do this is by introducing parallel programing concept into STRs identification [14-17]. And since, as far as we know, no tool with Parallel programing feature is as yet available [18, 19], a novel framework for the STRs using parallel string matching of the results compared with sequential approach has been devised.

STRs parallel string approach developed for scanning genomes to find repeats of any length, their exact position on chromosome and number of occurrence. It can accept large sequences and large number of files and directories up to GBs as input can be searched simultaneously on multicore processors. It divides the files and directories for different cores and run these files parallel then after it gives the files getting the output. Thus, the running time of the program is greatly reduced. The advantages over many other programs developed for STR identification includes its ability to do parallel programing using Multi-threading with multi core processingto search patterns repeated for a number of times of large input files and to give the exact position and number of occurrences of the pattern in the genome.

II. Methodology

Our goal is efficiently search STRs in large genomes of given pattern. The proposed methodology has two phases, Sequential and Parallel approaches. We model a basic algorithm which is giving better results than the previous algorithms like Boyer Moore, KMP and Brute force. In the first phase we apply on Sequential approach and later in the second phase the basic algorithm is to be applied on parallel string matching using Multi-threading with multi core processing concept. Finally we give a comparison between Boyer Moore, KMP and Brute force with our basic algorithm in sequential approach and then comparison between sequential approach and parallel approach of basic algorithm.

Methodology comprises of two phases

Phase 1:STR using Sequential String matching approach

The Sequential string matching approach uses the basic algorithm which is better than the previous algorithms and it processes the input file line by line from a document and run sequential manner to finding the repeats. The algorithm works with Improved Right Prefix concept, the below example demonstrate the working of the algorithm.

G	**G**	**C**	**C**	**G**	**A**	**G**	**A**	**A**	**G**	**A**	**T**	**A**
				G	A	T	C					

Figure 1: Example for Sequential pattern searching

Phase 2:STR Using Parallel String matching approach

In The second Phase "Short Tandem Repeats (STRs) Using Parallel String Matching" we apply the above basic algorithm to this concept and it processes the input file at first we take the input as a string or text. The required text that is to be searched is further divided into further small patterns or processes and all this patterns are passed on the basic algorithm and produce the result as number of occurrences of repeated pattern as shown in the Fig.2 below.

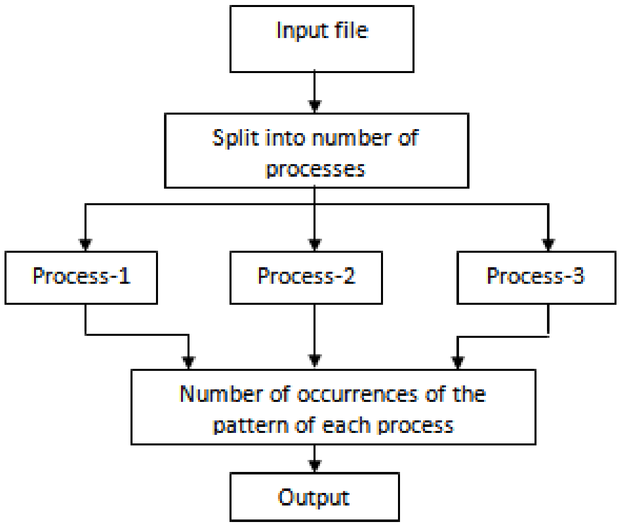
Input file
Split into number of processes
Process-1
Process-2
Process-3
Number of occurrences of the pattern of each process
Output

Figure 2: Parallel approach using Multi-Threading concept

Because threads can operate independently on different areas of an array for this algorithm, you will see a clear performance boost on multicore architectures compared to a sequential algorithm that would iterate over each process in the array. It uses the parallel execution of multi cores at a time which will results the speedup in the number of cores with minimal effort, because the Multi-threading takes care of maximizing parallelism.

The Flowchart of the Improved Right Prefix is shown in Fig 3.It .scans the genome sequence line by line for given input file and process the linefor given pattern length e.g. (ATGCATGCATGC)is a nucleotide with pattern count =3.If STR found then it identifies the position of the pattern and then increment the count value and search for next STR, if not then again reads the line of sequence. The process is continues until all the given patterns are found then it gives the number of occurrences of the pattern with line number and position.

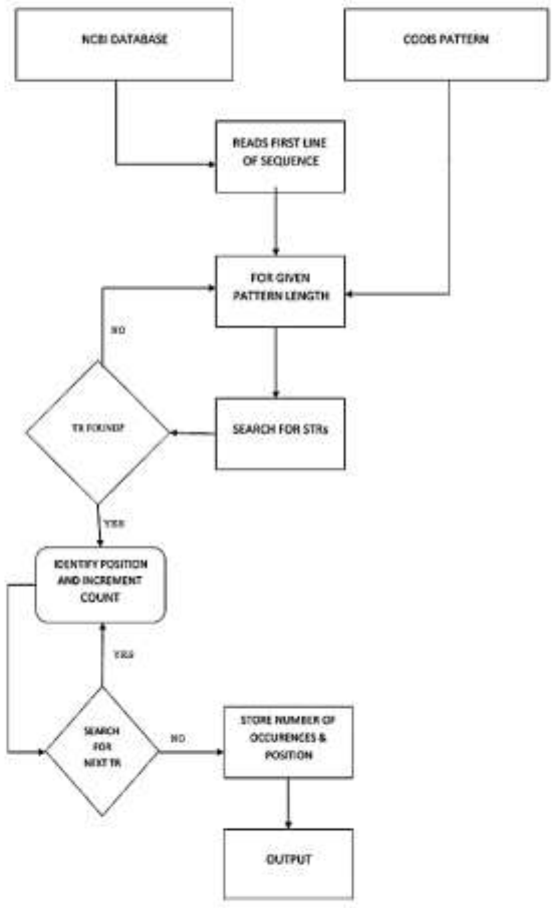

Figure 3: Flowchart of Improved Right Prefix algorithm

III. Results and Discussion

The sequential approach gives the better results by using the basic algorithm compared to previous algorithms Boyer Moore, KMP and Brute force with additional features like exact position with line number and number of occurrences. Then this basic algorithm reduces the time when we are applying in parallel string matching using Multi-threading with multi core processing,the results are illustrated in the histogram below. The sequential approach took 80ms, the parallel string matching using Multi-threading approach took 40ms.

The Fig 4 shows Execution time vs File size on sequential search withBrute Force, Improved Right Prefix Algorithm. This graph shows the performance difference between Brute Force,Improved Right Prefix algorithms. From the graph we clearly observe that IRP is better compared to Brute Force.

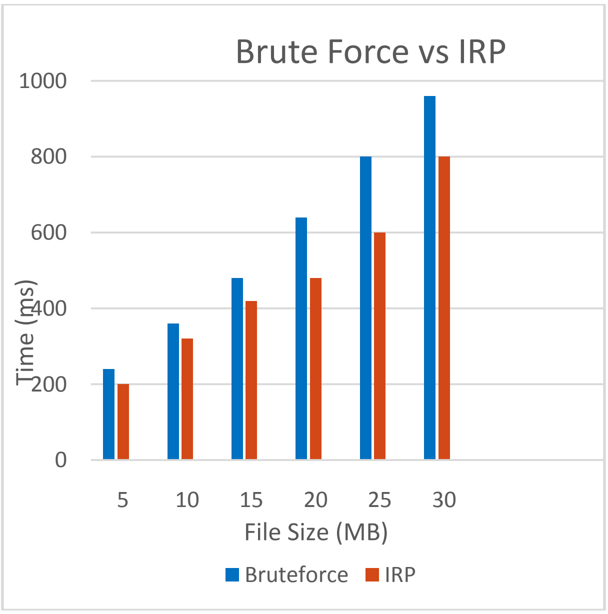

Figure 4: Histogram for Brute Forcevs IRP (Time vs File size)

The Fig 5 shows Execution time vs File size on sequential search with KMP, Improved Right Prefix Algorithm. This graph shows the performance difference between KMP,Improved Right Prefix algorithms. From the graph we clearly observe that IRP is better compared to KMP.The Fig 6 shows Execution time vs File size using Boyer Moore, IRP Algorithm on Sequential search. From the graph we clearly observe that IRP is much better than Boyer Moore.

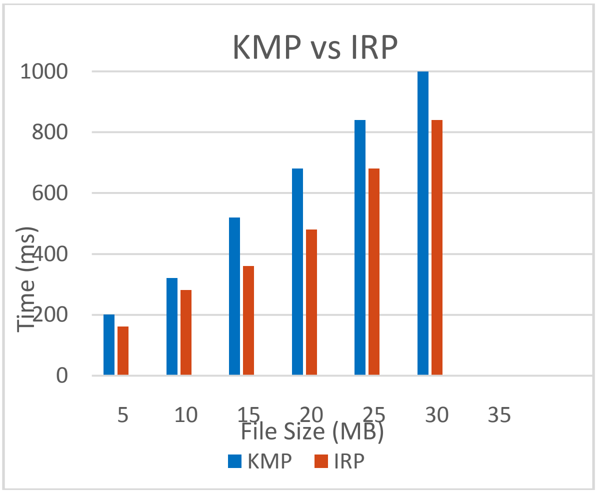

Figure 5: Histogram for KMPvs IRP (Time vs File size)

The most crucial stage in achieving a new successful system and in giving confidence on the system for the users that will work efficiently and effectively. The system will be implemented only after thorough testing and if it is found to work according to the specification. For testing our proposed system we will take the gene sequence data set, consists of the four nucleotides A, C, G and T (standing for adenine, cytosine, guanine, and thymine, respectively) used to encode DNA.

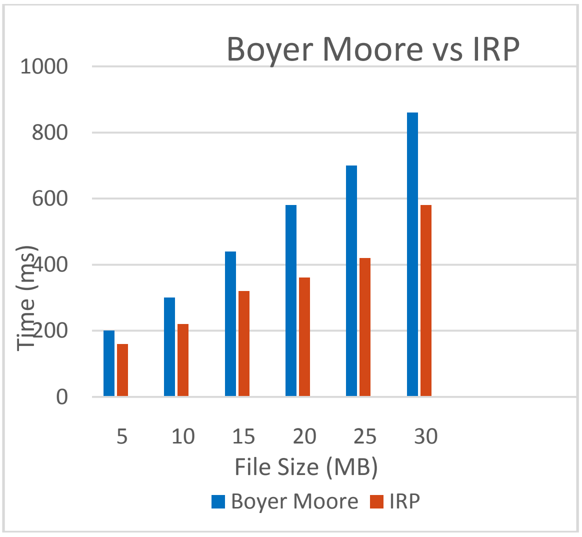

Figure 6: Histogram for Boyer Moore vs IRP (Time vs File size)

Therefore, the alphabet is O= {A, C, G, T}. The text is consisted of 7, 50,000 records. Our algorithm tested on different cores like 2, 4, 8, 12 etc., in the Fig 8 (Table). Here we put some achievements what we develop and observe, and finally our system shows that parallel approach is much better than sequential approach with multi core processor. The Fig 7 shows (Histogram) Execution time vs File size on sequential search and parallel search with Improved Right Prefix. From the Fig (4, 5, 6) we clearly observe that IRP is better compared to other approaches.

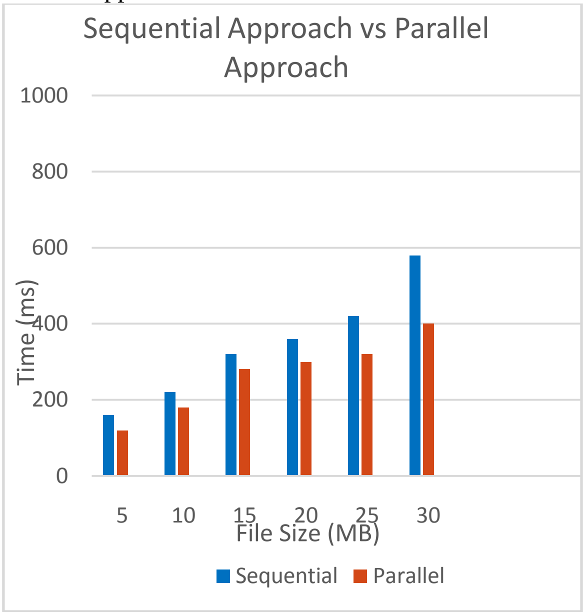

Figure 7: Histogram for Sequential vs Parallel approach

Number of Cores	Parallel Execution Time (MS)
2	11026
4	8329
8	4208
12	2876

Figure 8: Table of Parallel execution times with different cores

IV. Conclusion

In this paper we performed a comparisonbetween Knuth Morris Pratt, Boyer Moore and Brute force string matching algorithms with our improved right prefix algorithm in sequential approach based on the running time and in our tests with multicore processing, we used strings of varying lengths and texts of varying lengths. From the test results it is shown that the improved right prefix algorithm is extremely efficient in all cases. We apply this algorithm in parallel string matching using multi-threading concept to reduce the time by running on multicore processors. We conclude that STR using parallel string matching is the most efficient compared to earlier versions. As a future enhancement, we apply this algorithmto fork and join concept in JAVA SE7 of parallel string matching algorithm thereby finding the most efficient algorithm which can be used in many fields such as cryptography, molecular biology. Thus the problem of matching becomes easier.

References

[1] Angelika Merkel and Neil Gemmell (2008) *Detecting short tandem repeats from genome data: opening the software black box, VOL 9. NO 5. 355-366.*

[2] Gary Benson (1998) *Tandem Repeats Finder: A program to analyze DNA sequences Nucleic Acids Research, 1999, Vol_ 27, No. 2 573_580.*

[3] Karen Norrgard (Write Science Right) (2008) Forensics, *DNA fingerprinting, and CODIS. Nature Education 1(1):35.*

[4] Kathleen McNamara-Schroeder, Cheryl Olonan, Simon Chu, Maria C. Montoya, MahtaAlviri,ShannonGinty, and John J. Love (2005*) DNA Fingerprint Analysis of Three Short Tandem Repeat (STR) Loci for Biochemistry and Forensic Science Laboratory Courses* November 21, and in revised form, April 3, *Vol. 34, No. 5, pp. 378–383, 2006.*

[5] KoichiroDoi, TakuMonjo Pham H. Hoang, Jun Yoshimura, Hideaki Yurino, Jun Mitsui, Hiroyuki Ishiura, Yuji Takahashi, Yaeko Ichikawa, Jun Goto, Shoji Tsuji, and Shinichi MorishitaRapid(2014) *detection of expanded short tandem repeats in personal genomics using hybrid sequencing Bioinformatics 30 (6): 815-822.*

[6] TamannaAnwarandAsad U Khan (2006) *SSRscanner: a program for reporting distribution and exact location of simple sequence repeats***.**

[7] Jonathan L., "*Analysis of Fundamental Exact and Inexact Pattern Matching Algorithms*," Technical Document, Stanford University, 2004.

[8] ChintaSomeswararao, K Butchiraju, S ViswanadhaRaju, "*Recent Advancement is Parallel Algorithms for String matching on computing models - A survey and experimental results", LNCS, Springer, pp.270-278, ISBN: 978-3-642-29279-8,* 2011.

[9] ChintaSomeswararao, K Butchiraju, S ViswanadhaRaju, "*PDM data classification from STEP- an object oriented String matching approach", IEEE conference on Application of Information and Communication Technologies, pp.1-9, ISBN: 978-1-61284-831-0,* 2011.

[10] ChintaSomeswararao, K Butchiraju, S ViswanadhaRaju, "*Recent Advancement is Parallel Algorithms for String matching - A survey and experimental results", IJAC, Vol 4 issue 4, pp-91- 97,* 2012.

[11] Simon Y. and Inayatullah M., "*Improving Approximate Matching Capabilities for Meta Map Transfer Applications*," Proceedings of Symposium on Principles and Practice of Programming in Java, *pp.143 147,* 2004.

[12] ChintaSomeswararao, K Butchiraju, S ViswanadhaRaju, "*Parallel Algorithms for String Matching Problem based on Butterfly Model", pp.41-56, IJCST, Vol. 3, Issue 3,* July – Sept, *ISSN 2229-4333,* 2012.

[13] ChintaSomeswararao, K Butchiraju, S ViswanadhaRaju, "*Recent Advancement is String matching algorithms- A survey and experimental results", IJCIS, Vol 6 No 3, pp.56-61,* 2013.

[14] Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R. & V., S. "*PVM: Parallel Virtual Machine",* A Users Guide and Tutorial for Networked Parallel Computing, MIT Press. 1994.

[15] Grama, A., Karypis, G., Kumar, V. & Gupta, A. "*Introduction to Parallel Computing", Addison Wesley*, 2003.

[16] Leow, Y., Ng, C. &W.F.,W.. "*Generating hardware from OpenMP programs*", International Conference on Field Programmable Technology, *pp. 73–80,* 2006.

[17] Marr, D. T., Binns, F., Hill, D. L., Hinton, G., Koufaty, D. A., Miller, J. A. & Upton, M.."*Hyper-Threading Technology Architecture and Microarchitecture*", Intel Technology Journal, *pp. 4–15,* 2002.

[18] Donald Knuth; James H. Morris, Jr, Vaughan Pratt, "*Fast pattern matching in strings". SIAM Journal on Computing, pp-323–350,* 1977.

[19] Z. Galil, "*Optimal parallel algorithms for string matching," in Proc. 16th Annu.* ACM symposium on Theory of computing, *pp. 240-248,* 1984.