Intelligent Predictive String Search Algorithm Using Two Sliding Windows in Parallel Environment

Abstract : String matching strategies or algorithms provide key role in various real world problems or applications. These algorithms need to make less character comparisons and pattern shifts while searching for all occurrences of a pattern in a text. A few of its imperative applications are Bioinformatics , natural language processing and pattern recognition . Many string matching algorithms are existing and work efficiently with different applications in different life scopes; one of these algorithms is the Intelligent Predictive String Search Algorithm .this algorithm searches through a given text to find the first occurrence of a pattern without a preprocessing phase that included in many string matching algorithms to calculate the pattern shift values which lead less computations and uses simple rules during a match or mismatch of a pattern character using one sliding window. In this paper we updated the Intelligent Predictive String Search Algorithm three times resulting with three versions; in the first one we reversed the search direction to be from right using one sliding window while in second version we use two sliding windows to scans the text from both sides sequentially and finally we parallelize this version using real parallel environment.

Introduction : many algorithms that search for a certain pattern p of length m in a text t of length n exist, but they differ from each other in some aspects such as: Number of sliding windows used in searching process and Shift values. In this paper, we made an enhancement on the Intelligent Predictive String Search Algorithm , while keeping the shift values used in the original Intelligent Predictive String Search Algorithm as it is but we use two sliding window instead of one, the window size is equal pattern size (m), but In this case the two sliding window moves according to the same shift value rules. Comparisons are made between the new algorithm and the original Intelligent Predictive String Search Algorithm. The experimental results section showed that the new algorithm is faster than the others in case of a number of comparisons.

Related Work : In recent decades, many pattern matching algorithms have been developed and improved to meet the needs of various applications . Some of these algorithms, as previously described, used a single sliding window to search the text for a specific pattern, while others used two or more sliding windows. On the other hand, some of these algorithms require two phases, pre-processing phase to calculate shift values and searching phase. The shift values also varies from one algorithm to another, for example; the shifting values in the BoyerMoore algorithm (BM) in case of a mismatch (or a complete match of the whole pattern) depends on two pre-computed functions to shift the window to the right the goodsuffix shift and the bad-character shift. The pre-processing and searching time complexities of (BM) are O(m+|∑|) and Ω(n/m), O(n). Berry-Ravindran algorithm (BR) depend on the bad character shift function to determine the shift value in case of a mismatch and the searching phase make use of one sliding window from left to the right. The pre-processing and searching time complexities of BR algorithm are O(σ2) and O(nm) respectively. Two Sliding Window algorithm TSW enhanced (BR) by using two sliding windows instead of one, each of them equal to the length of the pattern n. One window aligned with the text from the left the other from the right and the both windows shifted according to bad character shift. In TSW, the best, worst and pre-process time complexity are O(m), O(((n/2-m+1))(m)) and O(2(m-1)) respectively . In order to minimize the number of comparisons, Enhanced Two Sliding Window algorithm (ETSW) made some modification on TSW. the modifications happened on the comparison process by using two pointers one from the left of the pattern and the other form the right of the same pattern. The same process applied to the two windows, the best, worst and pre-process time complexity are O(m/2), O(((n/2-m/2+1))(m/2+ and O(2(m-1)) respectively . ERSA uses two sliding windows the same as used in TSW. In addition to using (RSA) algorithm to calculate the shift values of the right pattern, some enhancement to calculate the shifting values for the left pattern was done to maximizes the efficiency of the searching process with O([n/(2* (m+4))]) time complexity in average case. The Intelligent Predictive String Search Algorithm, that we're going to develop in several stages have the following properties: It does not require pre-processing phase. It finds the first occurrence of a pattern in a text that consists of words separated by a blank space. It makes use of one sliding window to search the text from left It

uses two rules to make a shift namely alphabet-blank mismatch and alphabet-alphabet mismatch. We are going to explain the algorithm in detail in the next section.

METHOD:INTELLIGENT PREDICTIVE STRING SEARCH ALGORITHM

At each comparison this algorithm makes three main steps after aligning the leftmost character of the pattern P to the leftmost character of text T (Compare -> Predict -> Act).

Step1_Compare: At the beginning, the leftmost end of the pattern window with size (m) is aligned with the same end of the text. At each alignment of the pattern, the algorithm works on the portion of the text with size equal to pattern size (m) this is known as the text window. The comparison between the first character of the text from the left and the first character of pattern window from left is made, this comparison leads to either match or mismatch.

Step 2-Predict: In case of match is found the rightmost character of the pattern is compared with the rightmost character of the current window. If this leads to a match, the remaining characters are compared from right to left. In case of mismatch at any other position the pattern is shifted by (m) characters (window size).

In case of mismatch of the leftmost or the rightmost character of the pattern, the two rules Alphabet-Blank mismatch and Alphabet-Alphabet mismatch are taken placed depending on the type of mismatch. If t1 is a blank space, then pattern shifted by one position to right, however the next position might be a possible beginning of the pattern.

If t1 is different from pf1, then next character might be another character of the same word or it might be a blank. In case of the first possibility the pattern can be shifted by one position to the right while if the second possibility arises the pattern can be shifted by two positions. If t1 match p1, then check if tm is a blank or not. If tm is a blank this means that the text word currently checked is shorter than pattern, in this case the pattern is shifted by m positions towards the right if tm not a blank but differ from the corresponding pattern character and the next text character is blank, then pattern shifted by m positions towards the right.

Step 3-Act: In this step either a full match happened or the predicted shift value is taken place

- THE PROPOSED ADAPTATIONS ON THE INTELLIGENT PREDICTIVE STRING SEARCH ALGORITHM

In this paper we proposed three versions of Intelligent Predictive String Search Algorithm. The first two versions will enhance the performance of the algorithm under certain situations while the third version is shown to give better time and speedup of the original algorithm. The first variation which is named Right Intelligent Predictive String Search Algorithm which is suitable for searching for the last occurrence of the pattern in the text. The second variation called LR Intelligent Predictive String Search Algorithm, this algorithm makes use of two sliding windows to scan the text from its both sides right and left. Finally, the third variation refereed as Parallel Intelligent Predictive String Search Algorithm, in this algorithm they adapted the original Intelligent Predictive String Search Algorithm to work under a parallel environment which leads to enhancement in the performance in all situations. And its valuable in cases where we are interested in all occurrences of the pattern. We have to mentioned here that all versions used the Alphabet-Blank and Alphabet-Alphabet rules in case of mismatch exactly as used in the original algorithm. Next, we will describe each variation separately in details.

1-Right Intelligent Predictive String Search Algorithm

Right predictive will start by placing the pattern's right by the rightest position in the text, comparing the right most character of the pattern pm-1 with the right most character of the text tn-1, this may result with mismatch where the pattern is moved one position to the left if the text's character is a blank or two positions to the left otherwise. In case of matching, an attempt to find a total match starts by comparing the pattern and the text from the right of the pattern. In case of a mismatch the pattern slides by its length (m) to the left if the mismatch character in the text is a blank, or by (m+1) if the character to the right of the mismatch character in the text is a blank.

2- LR Intelligent Predictive String Search Algorithm (with two sliding windows) In this version the algorithm tries to get a match of the pattern inside the text by using two sliding windows to scan the text string from two directions; from left to right and from right to left. In

mismatch cases, during the searching process from the left, the left window is shifted to the right, while during the searching process from the right, the right window is shifted to the left. Both windows are shifted depending on Alphabet-Alphabet and Alphabet-Blank mismatch rules until the pattern is found or the windows reach the middle of the text.

3-A parallel version of predictive pattern matching algorithms

In this work, they follow decomposing the text and distribute it on different threads running at the same time to maximize speedup of the algorithm. This procedure can be applied on any of the Predictive algorithms: original, right predictive or LR predictive. The original predictive algorithm has been chosen for this work, but the same parallel algorithm can be applied to other versions of the predictive algorithm without difficulty. The parallel algorithm consists of three main parts: first, Text is decomposed into n parts (n is the number of threads), each Text part with the pattern is sent to a different computation thread. Second, each thread will run predictive pattern matching algorithm on its text part. Finally, results of simultaneous threads are collected and a proper output is presented.

Results:

. Analysis of results of the LR predictive algorithm

When considering the results, we notice that the original predictive algorithm will always show better performance in cases where the pattern is in the first part of the text file. This is clear in Figure8. In cases where the pattern is selected from the last part of the text file, it is clear that the two sliding window predictive algorithm will give a better performance in all cases. These results make sense, the original predictive algorithm will go all the way from the left to the right part of the text to find the pattern, whereas the LR predictive algorithm will try matching from one side at a time yielding a better performance. When considering the overall results, we conclude that in average, the two-window predictive algorithm will give better results in most cases. We have noticed that the performance of the two-window predictive algorithm will tend to degrade when that pattern is selected from the middle part of the text file, in this case the original predictive algorithm always gave better results.

Analysis of results of the LR parallel predictive algorithm

Both best results were obtained when the number of threads was two, which is equal to the number of CPUs in the machine. We can see that no further improvement can be done by increasing the number of threads, since the time tends to increase rather than decreasing after this peak point. This increase of time is due to communication overhead increase when number of threads increase.

In both experiments, one thread represents the sequential original predictive search algorithm. It is obvious that the parallel algorithm will give better results than the sequential original algorithm in all cases. As seen, the worst time in all recorded times is that of the case having one thread.

depending on the obtained results, we can say that the parallel version of the predictive algorithm will always give better results than the original version. Regarding the parallel predictive algorithm, the best search time and speedup will always be obtained when the number of threads is equal to the number of CPUs in the parallel machine.

CONCLUSIONS

In this present paper, three proposed versions of the Intelligent Predictive String Search Algorithm have been successfully implemented and tested. The experimental results of proposed LR parallel predictive algorithm method satisfy the lowest average search time and the highest speedup compared with the other approaches.