Improving the Efficiency of Exact Two Dimensional On-line Pattern Matching Algorithms

Charalampos S. Kouzinopoulos and Konstantinos G. Margaritis
Parallel and Distributed Processing Laboratory
Department of Applied Informatics, University of Macedonia
156 Egnatia str., P.O. Box 1591, 54006 Thessaloniki, Greece
E-mail: {ckouz,kmarg}@uom.gr

Abstract

This paper presents new efficient variants of the Baker and Bird and the Baeza-Yates and Regnier exact two dimensional pattern matching algorithms. Both the original algorithms and the variants are compared in terms of running time for different sets of data.

1. Introduction

String matching is an important problem in text processing and is commonly used to locate one dimensional patterns (strings) on texts. The expansion of imaging, graphics and multimedia required the use of pattern matching to higher dimensions leading to the two- and multi-dimensional pattern matching problem. The exact two dimensional pattern matching problem is defined in [2] as: let q be an alphabet, given a text array \( T[n \times n] \) and a pattern array \( P[m \times m] \), report all locations \([i,j]\) in \( T \) where there is an occurrence of \( P \), i.e. \( T[i + k, j + l] = P[k,l] \) for \( 0 \leq k, l \leq n \) and \( m \leq n \).

Several algorithms have been proposed by researchers for the exact two dimensional pattern matching problem and surveys and experiments on these algorithms have been already reported (The reader is referred to [9] for further details).

In this paper, new variants of the widely used Baker and Bird and Baeza-Yates and Regnier algorithms are presented and their efficiency is measured along the original algorithms.

2. Existing Algorithms

Baker [5] and independently Bird [7] introduced the first worst-case linear time algorithm for two dimensional pattern matching. The idea behind the algorithm is to run a finite automaton in order to perform a linear scan on the text. At each location of the text, the examination consists of two distinct steps: row-matching and column-matching. The row-matching step is used to locate rows of the pattern that match a substring of the text. If this step determines that a row \( P_i \) of the pattern occurs at a particular location, at the column-matching step must also be determined if the rows \( P_0, P_1, ..., P_{i-1} \) occur immediately above \( P_i \) in order to find out if the complete pattern appears as a sub-array of the text.

Both of these steps are performed using the Aho-Corasick algorithm [1]. Aho-Corasick treats each row of the pattern as a keyword and constructs a finite state pattern matching machine that can be used to process each row of the text in a single pass. The pattern matching machine then takes as an input a text’s row and returns the locations in which a row of the pattern appears as a substring or a failure if no such location exists. By using the finite state machine at runtime, the algorithm achieves a linear computation with \( n^2 + m^2 \) comparisons in the worst case.

The Baeza-Yates and Regnier algorithm [4] uses multiple string searching to locate a two dimensional pattern in a two dimensional text. It considers each pattern as \( m \) strings, each one of them with a length of \( m \) and the search is being decomposed into a one-dimensional multi-string search. Essentially, the searching is performed horizontally on only \( n/m \) rows of the text, since these rows cover all possible positions where the pattern may occur.

If a match is found starting on the \( k^{th} \) character of the \( i^{th} \) row of the text, then the \( m - i \) above and \( i - 1 \) rows below the current on the text are searched, starting from the same character position, to determine if a complete match occurs. The search is performed by using the Aho-Corasick algorithm as in the Baker and Bird algorithm. For the experiments presented in this paper, the simpler but practical version of the algorithm is used, a version that is using \( mn^2 \) character comparisons in the worst case and \( n^2/m + m^2 \) com-
parisons in the average case.

3. Plain-multi-string Variant

The Baker and Bird and Baeza-Yates and Regnier algorithms are very efficient in locating two dimensional patterns in two dimensional texts. However a number of disadvantages have been identified. Both algorithms are using the Aho-Corasick algorithm to perform the horizontal search on texts, an algorithm that needs preprocessing to construct the necessary structures before searching the text. For the same reason, extra space proportional to the size of the pattern is being used and the algorithms are rather complex without offering any significant performance improvement.

In this section a variant of the Baker and Bird and the Baeza-Yates and Regnier algorithm is presented in order to try and address these problems. Both variant algorithms are similar to the original algorithms, with the difference that the Aho-Corasick algorithm is replaced by the Plain-multi-string algorithm. Plain-multi-string is based on the Naive algorithm and is used to perform the horizontal search on a text. The function takes as an argument the text’s current row and column as well as the row of the pattern.

**Algorithm 1** Plain-multi-string algorithm on primary rows

1: for i = textcolumn to n - m + 1 do  
2:   k ← i  
3:   for j = 0 to m do  
4:     if pattern[patternrow,j] = text[textrow,k] then  
5:       k ← k + 1  
6:   else  
7:     break  
8:   end if  
9: end for  
10: if j = m then  
11:   return k - m  
12: else  
13:   j = 0  
14: end if  
15: end for  
16: return FALSE

When used on a primary row, Plain-multi-string returns the leftmost column c of the text’s row where a complete match of the string occurs, or reports a failure if no match is found. On subsequent searches on the same row, the column argument is shifted by one position to the right to avoid locating repeatedly the same match. When a match is reported on one of these primary rows of the text, the secondary rows are also examined. In that case, the algorithm returns TRUE if a match exists on the first m characters starting on column c of all the secondary rows or FALSE if a match is not found. The pseudocode for the two functions of the Plain-multi-string algorithm is presented in this section.

Both variant algorithms require only a fixed amount of extra space instead of the $m^2 + n$ extra space of the original Baker and Bird algorithm or the $m^2$ extra space of the original Baeza-Yates and Regnier algorithm while they are easier to be implemented, since the complicated Aho-Corasick algorithm is replaced with the much more compact Plain-multi-string algorithm. Our experiments showed that the variant of the Baker and Bird can be up to five times faster than the original algorithm while the variant of the Baeza-Yates and Regnier can be up to seven times faster than the original algorithm on some types of data.

**Algorithm 2** Plain-multi-string algorithm on secondary rows

1: for j = 0 to m do  
2:   if pattern[patternrow,j] = text[textrow,textcolumn] then  
3:     textcolumn ← textcolumn + 1  
4:   else  
5:     return FALSE  
6: end if  
7: end for  
8: return TRUE

4. Experimental Methodology

The experiments were performed locally on an Intel Xeon CPU with a 2.40GHz clock speed and 2 Gb of memory. The operating system used was a Debian Linux and during the experiments only the typical background processes ran. To decrease random variation, the time results were averages of 100 runs on different patterns and texts. All algorithms were implemented using the ANSI C programming language and were compiled using the GNU cc compiler with the “-O2” and “-funroll-loops” optimization flags. The implementation of some of the algorithms was based on code found on [6] and [8].

The data set used was a superset of the one used on [10] and [11]: The pattern had a variable size with $m = 4, 16, 32, 64, 128$ and $256$ while the text had two different sizes with $n = 1000$ and $n = 10000$. Both text and pattern were randomly generated with an alphabet size of $q = 2, 256$ and $1024$ to simulate bitmaps with different color depths.

5. Experimental Results

In this section, the performance in terms of running time of both the variant and the original algorithms is presented...
while is discussed the way the performance is affected by changes in text, pattern and alphabet sizes.

Figures 1 and 2 show the running time of the algorithms as plotted against the pattern size including any preprocessing. In each measurement, the data set used varied in terms of text, pattern and alphabet sizes. Figure 1 depicts the running time of the algorithms for text and pattern alphabets of size $q = 2$, 256 and 1024 against a text of size $n = 1000$ while Figure 2 displays the running time of the algorithms for the same alphabet sizes against a text of size $n = 10000$. Through examination of these Figures it can be seen that by varying pattern, alphabet and text sizes, the performance of the algorithms was affected in different ways.

As the pattern size increased, the running time of the Plain-multi-string variants of the Baker and Bird and the Baeza-Yates and Regnier algorithms slightly decreased; the horizontal search of the algorithms is performed on $n - m + 1$ characters and thus by increasing enough the size of $m$ the size of the text to be processed can be decreased significantly, especially when the size of $m$ and $n$ is relatively close (i.e. $m = 256$ and $n = 1000$). The original algorithms were not considerably affected by changes in the size of the pattern.

Using larger alphabet sizes affected both the original and the proposed variants of the Baker and Bird and the Baeza-Yates and Regnier algorithms, achieving up to three times gain in their performance when changing the alphabet from a binary to a larger sized one. This performance speed-up was due to the increase in the number of mismatches during the search of the text, a procedure that caused the search window to quickly slide towards the end of the text, skipping more character positions. The performance of these algorithms when a binary alphabet is used could be further improved by using bitwise operations in the implementation of the algorithms, achieving smaller sized data structures.
with extremely fast arithmetic operations between them\(^2\).

When a text of size \(n = 10000\) was used, the required time to search for all the occurrences of the pattern increased compared to the case where a text of size \(n = 1000\) was used, by an average of \(n^2\). This behaviour clearly shows that the size of the text is the dominant factor on which the performance of the algorithms mainly depends. As the size of the pattern increased, no speed-up was observed since the size of \(m\) was always much smaller than the size of \(n\).

Regardless of text, alphabet and pattern sizes it can be seen that the proposed variant of the Baeza-Yates and Regnier algorithm (plotted as “Baeza 2” in the Figures) had the best performance, followed closely by the proposed variant of the Baker and Bird algorithm (plotted as “Bird 2” in the Figures), the original Baker and Bird and the original Baeza-Yates and Regnier algorithm.

Figures 3 and 4 present the performance increase achieved by the Baker and Bird and the Baeza-Yates and Regnier algorithms when the Plain-multi-string variant is used. From these Figures can be seen that the variants of the algorithms had the greatest benefits in terms of performance when used on relatively small texts that use big alphabet sizes. With these types of data the variant of the Baeza-Yates and Regnier algorithm can be up to 7 times faster than the original algorithm while the variant of the Baker and Bird algorithm can be up to 5 times faster than the original algorithm. As the size of the text increases, the performance gain of the Plain-multi-string variant is smaller with an increase of up to 2.5 times when a large alphabet size is used comparing to the original algorithms. It is expected that for even larger text sizes the performance of the variants will tend to be similar to that of the original algorithms due to the dominance of \(n^2\).

Finally, Figures 5 and 6 show the results from the direct comparison between the original Baker and Bird and the original Baeza-Yates and Regnier algorithm as well as between their variants. When a text of size \(n = 1000\) was used, the original Baker and Bird algorithm was slightly faster than the Baeza-Yates and Regnier algorithm while when a text of size \(n = 10000\) was used the original Baeza-Yates and Regnier algorithm had an advantage in processing speed; on every type of data used the Baeza-Yates and Regnier variant was significantly faster than the Baker and Bird variant with a performance increase of up to 1.7 times.

6. Conclusions

In this paper a new variant of the Baker and Bird and the Baeza-Yates and Regnier was proposed. Both the variant and the original algorithms were compared in terms of running time and as was discussed in the previous section the variant algorithms had a performance improvement of up to seven times in some cases comparing to the original algorithms.

Baker and Bird had a linear behaviour on pattern size changes and was quite efficient on both small and larger data sets but on the other hand it had the need for a considerable amount of extra space \((n+m^2)\). The proposed variant improved the performance of the algorithm when a small sized text was used up to a factor of five while having the need for only a fixed amount of extra space.

The Baeza-Yates and Regnier algorithm performed searches for patterns fast, independently of pattern, text and alphabet sizes. The proposed variant had a significant effect on the speed of the original algorithm, achieving in some cases up to seven times better performance. The variant also had the need for only a fixed amount of extra space instead of \(m^2\).

Future research in the area of two dimensional pattern matching could focus on further speeding up the existing algorithms. By using more advanced optimization techniques like bitwise operations and loops reorganization, the search for a two dimensional pattern on a two dimensional text could become more efficient. Moreover, the additional speed up of the algorithms presented in this paper by parallel processing them could be researched, especially when large data are involved (i.e. aerial imaging) or in time-critical applications. Finally, it would be interesting to examine the performance of the Plain-multi-string variant when additional types of data are used, like real-life data and non-random data with varying percentage of hits and misses.

References


---

\(^2\)One such algorithm that is using bitwise operations is the Shift-Or \([3]\).
Figure 3. Comparison of original Baeza over Baeza variant for $n = 1000, 10000$

Figure 4. Comparison of original Bird over Bird variant for $n = 1000, 10000$

Figure 5. Comparison of original Bird over original Baeza for $n = 1000, 10000$

Figure 6. Comparison of Bird variant over Baeza variant for $n = 1000, 10000$

