Research of BM Algorithm in Character Match

Zhu Baofeng1, a, Zhao Bing2,b
1Henan Institute of Education, China
2ZhongZhou University, China
a37394008@qq.com, b781705841@qq.com,

Abstract

Character match is more and more applied in computer application field. BM algorithm belongs to an algorithm based on bad sign and good suffix rules. It has high match efficiency, which is widely applied. In the paper, some application fields of BM algorithm are firstly analyzed and discussed. Several character match algorithms similar to the BM algorithms are analyzed, then basic concepts of BM algorithm are analyzed in details, calculation method of bad sign shift table and good suffix shift table is designed and implemented, pseudo-code of the algorithm is constructed through the calculation method. A group of data is randomly selected for example analysis finally. Test results show that improved BM algorithm significantly shortens the character matching time and greatly improves efficiency of character match.

Keywords: character match; BM algorithm; bad sign; good suffix; efficiency.

Introduction

Application of processing non-numerical data is growing rapidly with rapid development of computer application technology[1,2]. Character matching problem is particularly important, which is related to intrusion detection, information retrieval, search engines and other fields. Network applications are globalized gradually with rapid development of Internet, and increasing expansion of network scale. Hacking attack and intrusion also occur continuously. It is difficult for traditional firewall technology to separately safeguard network security. Network intrusion detection system, as a proactive security protection technology, has become a hotspot researched in network security field. Character matching algorithm is frequently applied in the system for intercepting unknown request, thereby playing a protective role.

Character match refers to the first occurrence of searched pattern string in text string. Currently, there are many character matching algorithms, such as brute force algorithm, BM algorithm, Horspool algorithm, Wu-Manber algorithm, etc[3,4]. Character matching algorithms can be divided into three categories based on different character scanning strategies[7]: 1) each pair of corresponding characters in the pattern and text is scanned from left to right. If the characters are not matched, the mode can be shifted rightwards for one grid in next trying. Brute

force character match belongs to such algorithm, and its average efficiency belongs to Θ (n * m). 2) The maximum suffixes of text string and pattern string are matched from right to left, BM algorithm and Horspool algorithm belong to this category, the worst efficiency of BM is linear, and Horspool is a simplified version of BM. 3) Maximum substring of text string and pattern string is matched from right to left[5].

BM algorithm has been considered as the most effective character matching algorithm. BM algorithm [1] is adopted for open source intrusion detection system Snort. In the paper, basic idea of BM algorithm, calculation of bad sign shifting table and good suffix shifting table are analyzed, performance of BM algorithm is analyzed through an example, thereby improving BM algorithm and increasing efficiency of algorithm to some extent.

Basic Idea of BM Algorithm

Definitions. Definition 1: Set Σ as alphabet, T, P∈Σ +, wherein text string T = t0t1t2 ... tn-1, pattern string P = p0p1p2 ... pm-1, m <n; Definition 2: c∈Σ and c∈T, c is a bad sign, thereby resulting in mismatch of corresponding character between pattern string P and character string T; d is the shifting distance of pattern string corresponding to text string; k∈N is the maximum suffix for matching text string and pattern string[6].

Thought of BM Algorithm. (1)The text string T is aligned with pattern string P on the left, corresponding character pairs of text and mode are compared from right to left since pm-1, If p0p1p2…pm-1 are matched with titi+1ti+2…ti+m-1 in T one by one, P occurrence in T can be searched, which returns to T. If pattern P exceeds text T, P does not appear in T, which returns to -1. (2) Primary comparison between the rightmost character pm-1 in mode string P and corresponding character c of text string T fails, mode string P should be moved rightwards for sufficient distance as long as possible[2], thereby reducing movement frequency and comparison frequency, and lowering algorithm complexity. If c is not included in former m-1 character of P, the maximum distance of P shifting is len(P)=m. If c appears in the former m-1 character of P (appearance frequency may be >1), P can be moved rightwards then, thereby the rightmost c in P can be aligned with c in T, and P's shifting distance is related with c. Rightward shifting length of mode P is expressed with t (c):

$$t_1(c)=\begin{cases} m(c \notin P[0,1,2...m-1]) \\ m-j \end{cases} . \quad (1)$$

c=pj, and c is the rightmost c in former m-1 character of mode P. (3) Before pattern string P and text string T encounter bad sign c, k (0 <k <m) characters have been successfully matched, leftward shifting length d of pattern string P should be based on the following two rules: bad sign shifting rule and good suffix moving rule.

Bad sign shifting rule: This rule should adopt bad sign c as reference. If c is not in P, P should be shifted to just skip character c. The movement length should be represented as m-k. If c exists in the mode, the rightmost c of m-k character in front of P can be aligned with c in T, please refer to equation 1, P shifting distance is represented by t(c)-k. When k is large enough corresponding to t(c), t (c) -k <0 can be caused, P can be moved rightwards for one position by referring to brute force algorithm. Bad sign shifting table can be calculated aiming at each character in alphabet Σ. Bad sign shifting distance is represented as the follows:

$$d_1=\max\{t(c)-k,1\};$$

(2)

Good suffix shifting rule: k characters which have been matched between pattern string P and text string T are called good suffixes, and denoted as suff (k). If one longest character combination v(len(v)<k) can be searched in the former m-k characters of pattern P, v(len(v)<k) is suffix of suff(k), P should be shifted for aligning the rightmost v of the former m-k characters with v in suff(k), and v is related with k. Distance of good suffix shifting is represented as follows:

$$d_2=\begin{cases} v & \\ m & (v \notin \text{suff}(k)) \end{cases}$$

(3)

There is a longest prefix v in the distance (suff(k) between *v* and suffix *suff(k)*. The $d=\max(d_1, d_2)$ is regarded as the maximum rightward shifting distance of P. Bad sign shifting table and good suffix shifting table are constructed in advance during implementation of algorithm, thereby reducing value d searching and computing frequency under different effects of c, v and k, and improving efficiency.

Implementation of BM Algorithm Improvement

Improvement Idea of BM Algorithm. BM improvement ideas are proposed through analysis and practical application of original BM, thereby accelerating backward jumping magnitude after match failure. Specific improvements are shown as follows. BM value from t [0] to t[n-1] character in t should be firstly obtained in BMGJ module. BM value of all characters is set to be n+ 1 at the beginning. Then, BMGJ value of character in the pattern string is calculated, namely position of the character in the pattern string from the rightmost side. Matching mode from right to left is adopted during character string match by sBMGJ. When it is discovered that corresponding characters of the jth characters in text and pattern strings are not equal, it is assured that the pattern string should be shifted rightwards for at least one character. Shifting distance can be decided according to next character of character in the first comparison. If pattern string length is 6 characters, the characters should be matched by the algorithm from right to left. The sixth character in the text should be compared in the first match, and they are not matched as a result. Therefore, the substring should be moved backwards. The displacement should be decided according to the last character

("r") aligned with the compared string in BM algorithm. The displacement can be obtained according to the ("i") closely behind the current substring in the improved algorithm. Since shifting is necessary due to failure of last match, the shifting distance should be at least one character. If next character is in the pattern string, the shifting distance should be decided according to the position of pattern string. If it does not occur in the pattern string, comparison is not necessary, therefore, we can move to next character of the character for comparison. The whole process for the method showed in figure 1.

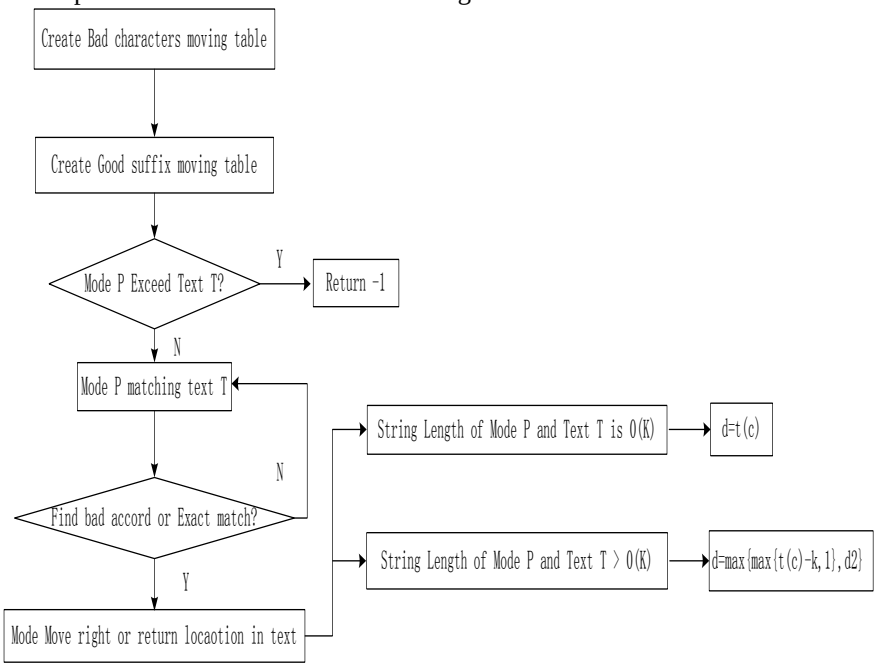

Figure 1 Program Flow Chart Exhibition

Analysis of Good Suffix Rule. Theoretically better rules do not play much role in practical application, which will affect performance of algorithm on the contrary. Disadvantages of good suffix rule are analyzed in details next: (1) Pretreatment is necessary for applying good suffix rule. Pretreatment process of good suffix rule is actually a pattern matching process, suffix substring is searched in pattern string. The process can be more and more complex and consumed time can be longer and longer with pattern string length increase. The process has high time complexity, which is far longer than linearity. (2) It is impossible to form good suffix in the comparison process. Good suffix rules are different from bad character rules, which can be used after more than one characters are matched. However, some characters at the end of pattern string are impossibly matched with corresponding characters in the target string. In addition, it is also related with occurrence possibility of pattern string ending characters in the target string. (3) If the ending characters in pattern string have high repeating possibility, good suffix

application rate will be low. Good suffix rules can not be applied in the matching process if suffix is available. If the ending character has high repetition rate in the pattern string, or reappearance distance is short, it is difficult to use good suffix rule.

Realization of BM Algorithm Improvement. Heuristic methods are used to improve the efficiency of BM algorithm and its improved algorithm. Memory entries for accelerating character match 'jumping' are created at the initial stage of algorithm. Next shifting distance of the matching window can be determined through constantly searching the table in the algorithm matching stage. Key step length of several mode matching algorithms and memory access comparison condition are analyzed. Memory access overhead of BM algorithm is relatively small. Therefore, algorithm table searching contents and comparison judgment sequence should be suitably adjusted besides key step length and key step length probability. Table searching frequency and access frequency demanded for match window shifting key step length should be lowered for prominently improving actual implementation rate of the algorithm. BM algorithm can be improved and optimized accordingly.

If match from next character is considered in snort BM algorithm, the displacement produced in the initial calling character matching process can be shifted for at least two more characters compared with displacement of original algorithm, implementation of single character match is only increased once. Displacement of the string can be accelerated under the condition that the tested character text length is fixed, thereby increasing the operation speed of BM algorithm. Therefore, optimization treatment on the basis of BM algorithm can be considered. How to use next byte in T text as matching foundation should be considered in improved algorithm. If the algorithm is adopted at the beginning of match, the following conditions can be achieved: if P character string is firstly compared in T text, accurate match is not available. Therefore, test accuracy should be ensured at the beginning of match, corresponding characters in T text should be used for comparison. When next character is selected, it can be conducted under the condition that some characters on the right have been matched. If matching character is discovered on the right of unmatched characters, next character and the last character in P should be adopted. Shift algorithm in source procedure should be continued, therefore character displacements can be compared conveniently. Since shift algorithm may be larger displacement in the matching process, the maximum displacement should be compared.

Test Result of Improved BM Algorithm

Experimental Conditions and Experimental Data Show. Randomly selected pattern string P is algorithm, Σ t (c) value of each character in character set $\sum$ should be calculated according to equation 1. Bad symbol shifting table should be constructed as shown in table 1.

Table 1 Bad Symbol Shifting Table

c	a	l	g	o	r	i	t	h	m	Other Symbols

t1(c)	8	7	6	5	4	3	2	1	9	9

The d_2 value under each value of k(0<k<m) can be calculated according to formula 3. Each value of k should be traversed. Character combination v is not available in mode P, and suffix shifting table should be constructed as shown in table 2.

Table 2 Good Suffix Shifting Table

k	Mode P	d2
1	algorithm	9
2	algorithm	9
3	algorithm	9
4	algorithm	9
5	algorithm	9
6	algorithm	9
7	algorithm	9
8	algorithm	9

Analysis of Concrete Examples.

T: this_is_boyer_moore_algorithms

P: algorithm

Character ‘b’ in text string is compared with the rightmost character ‘m’ of pattern string, they are not matched for the first time, and b is bad sign. The shifting distance is 9 according to Table 1, and the comparison frequency is 1.

T: this_is_boyer_moore_algorithms

P: algorithm

The character "r" and character "m" are compared, they are not matched for the first time, r is bad symbol, shifting distance is 4 and comparison frequency is 1 according to table 1.

T: this_is_boyer_moore_algorithms

P: algorithm

The character "l" and character "m" are compared, they are not matched for the first time, 1 is bad symbol, shifting distance is 7 and comparison frequency is 1 according to table 1.

T: this_is_boyer_moore_algorithms

P: algorithm

Table3 showed the times for the algorithms taken.

Table 3 Improved Performance Data Table

Name	A	B	C	D	E	Average
Before improved	5.521	5.212	5.748	7.921	5.311	5.9426
After improved	4.321	4.572	4.365	5.54	4.356	4.6308
Residual value	1.2	0.64	1.383	2.381	0.955	1.3118

Conclusion

In the paper, application of character matching algorithm application in various fields is analyzed firstly. Features of existing character matching algorithm are briefly introduced, definition and basic concepts of BM algorithm are deeply analyzed. Basic features of good suffixes are studied on the basis. Basic concepts of improved BM algorithm are proposed, mechanisms of improved BM algorithm are designed and realized, finally practicability and test effects of the algorithm are proved through concrete example. In the example, scale n of text string is 30, and the scale of pattern string is 9. Algorithm speed is very fast. The efficiency of inputting text string and character string belongs to Θ (n). Example is not related to good suffix shifting table during calculation of P shifting step length, which is simplified version of BM algorithm. The longest prefix of mode is available under rare cases in practical application. Bad sign shifting rule in BM algorithm is more applied, which is more applicable during treatment of natural language strings. The paper has the following shortcomings that overall performance of all algorithms is not comprehensively compared and analyzed, effective algorithm performance criterion and design models are not proposed. Therefore, we will test the algorithm performance more quantitatively through model establishment based on designing scientific performance indicators in the future.

References

[1]L.J. John, M.L.Padgett: IEEE Transactions on Neural Networks Vol.10(3), p.480-498.

[2] R.S. Boyer, J.S.Moore: Communications of the ACM Vol.20-26 (2007), p.762-772.

[3] A.V.Aho, M.J. Corasick: Communications of the ACM, Vol 16-20(2006), p.333-340.

[4] D. Nam: String matching algorithms in finding domain name and IP address (Springer Netherlands, Berlin,2010).

[5]O.Hiroyuki: IEICE Transactions on Communications Vol.85-87(2011),p.107-115.

[6]M. J.Osborne, Rubinstein A.A course in game theory(MIT Press，Cambridge,1994).

[7]G.H.Han,C. Zeng: Journal of computer application Vol.33(8), p.2379-2382.