Fuzzy Pattern Matching Algorithm for Location Based Approximate Strings

Anu Sebastian, Joby George

Abstract— String matching is a classical problem in computer science. The problem of string matching has been studied extensively due to its wide range of applications from Internet searches to computational biology. Applications may require exact or approximate string matching. There are many approximate pattern matching algorithms proposed in the literature. They mainly focus on solving the k-mismatch problems and k-difference problems. K-mismatch problems find all occurrences of a short pattern in a long text string with at most k mismatches. But these mismatches can be anywhere in the pattern. But in some applications it is essential to find the fuzzy patterns varying only in some specific positions of the pattern. For example finding Transcription Factor Binding Site along the DNA is an application of such type of fuzzy pattern matching. The paper proposes an approximate pattern matching algorithm that allows only position based variation in the pattern. The algorithm can be used to determine the potential Transcription Factor Binding Sites along the DNA and for similar applications.

Index Terms— Algorithm for bioinformatics, pattern matching, fuzzy pattern matching, position specific approximate pattern matching, Transcription Factors, Transcription Factor Binding Site, Boyer–Moore algorithm.

1 Introduction

String matching is a technique to discover pattern from the specified input string. It is the problem of finding all valid shift with which a pattern P occurs in text T[1]. The problem of string matching has been studied extensively due to its wide range of applications from Text editors in computing machines, Database queries, network intrusion detections system, Bioinformatics and Cheminformatics, wide window pattern matching, music content retrievals, MS-word spell checker, matching DNA sequences, language syntax checker, digital libraries, search engines and many more. In string matching approximate pattern matching is considered as a challenging problem. It is a recurrent problem which is applied in text searching, pattern recognition, computational biology and signal processing applications [13]. Approximate pattern matching is also called fuzzy pattern matching.

Approximate pattern matching operations usually consume a huge amount of computational resources. K-mismatch problem and the k-diff erence problem are the main two variations of the approximate pattern matching problem. In both the problems, a short pattern string P = p1p2 ··· pm and a long text string T = t1t2t3 ··· tn over an alphabet Σ, and an integer k are given. The Σ may be a typical human alphabet like the letters A through Z. In some applications binary alphabet Σ = {0,1} or DNA alphabet Σ = {A,C,G,T} are also used. The k-mismatch problem is to find out every occurrences of P (pattern) in T (text) with at most k mismatches permitted. Whereas the k-diff erence problem finds all substrings of T (text) with edit distance at most k to P (pattern). In literature there are many algorithms to solve these problems.

• *Anu Sebastian, Mar Athanasius College of Engineering, Kothamangalam, India, anoos87@gmail.com*
• *Joby George, Mar Athanasius College of Engineering, Kothamangalam, India, jobygeo@gmail.com*

The naive algorithm calculates the Hamming distance for every alignment of the pattern P in the text T in time O(nm) for string matching with mismatches, The motivation for approximate string matching comes from low quality of text, heterogeneousness of databases, spelling errors in the pattern or text, searching for foreign names and searching with uncertainty [13]. In some application we need to find fuzzy patterns from a long string but the uncertainty is applied only to some fixed positions in the pattern. Whereas the remaining positions in the pattern is fixed, that is it can have only fixed alphabet from Σ. For example if the pattern is '*AT', where '*' can be replaced with any alphabet in {B, C, E, F, H, M, R} to form {BAT, CAT, EAT, FAT, HAT, MAT, RAT} and the second and the third positions have the fixed alphabet 'A' and 'T' respectively. The first position is not allowed to have any alphabet other than from the set {B, C, E, F, H, M, R}. Finding all the occurrences of pattern '*AT' from a long string is therefore an approximate pattern matching problem with position based approximation. Here the approximation is applied only to the first position, and the first position can have only an alphabet from the set {B, C, E, F, H, M, R}. Remaining positions in the pattern is fixed. So this type of pattern matching comes in between the fixed pattern matching and approximate pattern matching. As per our knowledge there is no solution addressing this specific type of fuzzy pattern matching problem.

2 Specification of Position Based Approximation

Position specific approximation in the pattern can be specified using a position based presence matrix. The position based presence matrix is a |Σ| × l matrix. Where |Σ| is the number of alphabet in the Σ and l is the length of the pattern. The position based presence matrix for the pattern '*AT' is shown in the table 1. Here Σ contains the letters A through Z.

TABLE 1
POSITION BASED PRESESNCE MATRIX FOR THE PATTERN '*AT'

	*	A	T
A	0	1	0
B	1	0	0
C	1	0	0
D	0	0	0
E	1	0	0
F	1	0	0
G	0	0	0
H	1	0	0
I	0	0	0
J	0	0	0
K	0	0	0
L	0	0	0
M	1	0	0
N	0	0	0
O	0	0	0
P	0	0	0
Q	0	0	0
R	1	0	0
S	0	0	0
T	0	0	1
U	0	0	0
V	0	0	0
W	0	0	0
X	0	0	0
Y	0	0	0
Z	0	0	0

Here the entries corresponding to the column '*' have non zero value when the alphabet is from the set {B, C, E, F, H, M, R}. Similarly 'A' and 'T' have corresponding entry as 1 and rest of them as 0.

This type of pattern matching with uncertainty is applicable in many areas. The matching algorithm is discussed from the context of bioinformatics, but applicable to all fields. Finding Transcription Factor Binding Site along the DNA is an application of such type of fuzzy pattern matching. Transcription Factors (TFs) are proteins that does DNA regulation mechanism. All living organisms are composed of cells. Cells of different cell types may differ greatly in their morphology function according to the tissue they form. For example, the axons of the neuronal cells in human can be over one meter long, the skeletal muscle cell can span tens of mm, whereas the size of a white blood cell is about 7μm in diameter. But the genetic information encoded in the nucleus of these cells is nearly identical. The differentiation of the cell is strongly controlled through the regulation of gene expression. One such regulation mechanism is done byTFs [2].

TFs bind to the DNA molecule to control the expression of their target genes. One of the distinct characters of transcription factors is that they have a DNA-binding domain that recognizes a short specific DNA sequence. These short DNA sequences are usually different for distinct transcription factors and are called transcription factor binding sites (TFBS) [3]. Every transcription factor is able to bind not only to a single DNA sequence but to a variety of DNA sequences that share a core structure [4]. It is represented by the binding motif. Table 2 shows bound sequences for a TF. In all those sequences the core structure remains the same as '*AA*ATGGC*G*'.

TABLE 2
BOUND SEQUENCE

SITE	SEQUENCE
1	CAAGATGGCGGC
2	GAAGATGGCGGC
3	GAAGATGGCGGT
4	CAAGATGGCTGT
5	CAAAATGGCCGC
6	AAAAATGGCGGC
7	CAAGATGGCCGC
8	AAAGATGGCTGC
9	CAAAATGGCTGC
10	CAAGATGGCCGT

Recognizing potential TFBS along DNA is finding the locations in DNA that satisfies the binding motif. The binding motif is an approximate pattern that allows variation in particular positions. A pattern matching algorithm that can accommodate position based variation is essential to find the potential TFBS. In case of TFs the variation is specified using a Position-Specific Frequency Matrix (PSFM) like the position based presence matrix introduced earlier. Fuzzy pattern matching algorithms like Tarhio and Ukknen [5] allows k-mismatches in the pattern. But this mismatches can be anywhere in the pattern. Whereas TFBS share a core structure in which the pattern remains constant in some positions and it can vary in other positions. So an algorithm that allows variations in specified positions can be used locate the potential TFBS along the DNA. The paper proposes an algorithm based on the Boyer-Moore technique [6] to allow position based variation in the pattern. The same method can be used along with the KMP (Knuth–Morris–Pratt) string matching algorithm to find all the occurrences of the position based vague pattern from a long text in polynomial time [14].

3 Transcription Factors and Their Binding Affinity

Binding of transcription factors to the DNA is to some extent stochastic and depends on the biophysical properties of the DNA sequence [7]. The binding sites for a transcription factor share a common core structure. These patterns of transcription factor binding site for a single factor can be represented with a Position-Specific Frequency Matrix (PSFM). PSFM has row entries for each symbol of the DNA alphabet (nucleotides A=adenine, C=cytosine, G=guanine and T=thymine) and column entries for each position in the pattern. That is Position-Specific Frequency Matrix is a 4×l matrix where l is the length of the transcription factor. The cell entries of a PSFM are calculated as relative frequencies of each nucleotide at each position in the pattern.

Visual representation of this is done using a sequence logo. In such a graphic, a sequence logo shows stacked nucleotide symbols of heights proportional to their information content at the respective position. The higher the preference of a nucleotide is in the PSFM at a given position, the higher is the probability of corresponding letter at that position in the sequence logo [8]. The PSFMs, and the DNA-bindings sequences are stored in several databases such as TRANSFAC [9], JASPAR [10] and UniProbe [11]. Figure 1 shows the sequence logo for the transcription factor NFIC::TLX1 (id- MA0119.1) from the JASPAR database. The Position Frequency Matrix for the same is given in the figure 2.

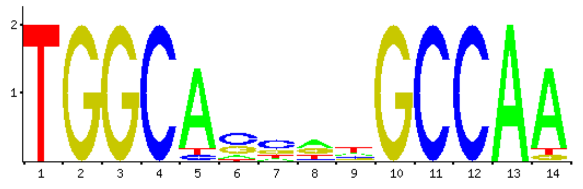
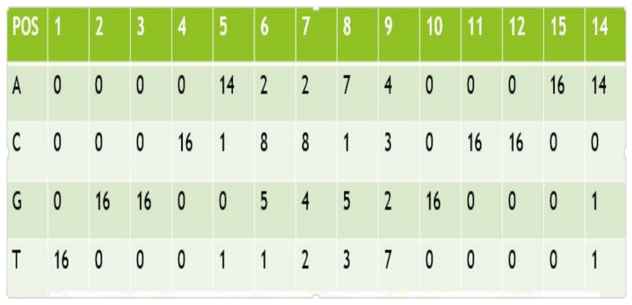


Fig. 1. Sequence logo for TF NFIC::TLX1



POS	1	2	3	4	5	6	7	8	9	10	11	12	15	14
A	0	0	0	0	14	2	2	7	4	0	0	0	16	14
C	0	0	0	16	1	8	8	1	3	0	16	16	0	0
G	0	16	16	0	0	5	4	5	2	16	0	0	0	1
T	16	0	0	0	1	1	2	3	7	0	0	0	0	1

Fig. 1. PFM for TF NFIC::TLX1

The sequence logo represents the binding motif for the transcription factor which is an approximate pattern. From the sequence logo or from the PFM it is clear that the vague pattern is 'TGGC*****GCCA*', where the'*' in the 5th position can have {A, C, T} not G. 6th position can have any nucleotide {A, C, G, T}. Last position (14th) can have any nucleotide from {A, G, T} not C similarly in all the positions. Height of the letter in a position or the value in the PFM represents the probability of occurrence of that nucleotide in the corresponding position.

4 Position based Approximate Pattern Matching

String matching is a vital problem. It is highly recommended to have fastest algorithms in different application. Boyer–Moore algorithm is one of the efficient algorithms for exact pattern matching [6]. It is considered as the standard benchmark for pattern matching algorithms. Boyer-Moore algorithm aligns P (pattern) with the T (text) successively, then checks whether P matches the opposite characters in T. When the scanning phase is complete, P is shifted to right relative to the text T. Boyer – Moore algorithm contains three clever ideas:

- The right to left scan
- The bad character rule
- Good suffix rule

We modified the Boyer–Moore algorithm to allow position specific variation in the pattern to find out the Transcription Factor Binding Site based on the Position Frequency Matrix. Using position based presence matrix the vague pattern from the text can be found out in any other applications.

4.1 Bad character rule modified

The pattern P and the text T are matched from right to left. The bad-character rule considers the alphabet in T at which the mismatch occurred. Then the next occurrence of that alphabet to the left in P is found out, and a move which takes that occurrence in line with the mismatched occurrence in T is made. If the mismatched alphabet does not occur to the left in P, entire P is shifted past the point of mismatch. The following example clarifies the statement.

```
    0                  1                  2
    1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4
T:  T T A A T A A T G A G C C T A T A G C A A C G T

P:  C A T G C C A T

            C A T G C C A T
```

The Boyer–Moore algorithm scans from the right end. The mismatch occur at the 3rd position from right of the pattern (in the 6th position of the pattern) and a shift of distance 4 is made based on the Bad character rule. Since the mismatched character in the text (T) is 'A' and 'A' occur in the 2nd position of the pattern (P). In our work in order to accommodate position based variation in the bad character rule, the Position Frequency Matrix (PFM) is incorporated in the preprocessing of the bad character rule.

A 2D table is constructed which is indexed first by the index of the alphabets in DNA (A, C, G, T) and second by the index i in the Transcription Factor Binding Motif (pattern). Table is made with entries as the shift distance. 2D table for the TF NFIC::TLXI (Fig1) based on the PFM (Fig2) is given in the table3. Where shift distance is the i-j (j<i), where j is the highest index of the occurrence of the corresponding DNA alphabet based on the PFM (ie, highest index where the entry>0 for that alphabet in PFM). For example the entry for T×4 is 3 because from the 4th position in the pattern, a shift distance of 3 to left side is needed to reach position with non-zero entry for T in PFM.

TABLE 3
BAD CHARACTER RULE PRE PROCESSING FOR TF NFIC::TLXI

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
POS	T	G	G	C	*	*	*	*	*	G	C	C	A	*
A	1	2	3	4	5	1	1	1	1	1	2	3	4	1
C	1	2	3	4	1	1	1	1	1	1	2	1	1	2
G	1	2	1	1	2	3	1	1	1	1	1	2	3	4
T	1	1	2	3	4	1	1	1	1	1	2	3	4	5

While performing the matching, bad character rule uses the entry in the table as the shift distance.

4.2 Good Suffix rule modified

The good suffix rule is: For a given alignment of P and T; if a substring s of T matches a suffix of P, but a mismatch arises at the subsequent comparison to the left. Then find, if there exists, the right-most copy s' of s in P such that s' is not a suffix of P and the character to the left of s' in P differs from the character to the left of s in P. Shift P to the right so that substring s' in P aligns with substring s in T. If s' does not exist, then shift the left end of P past the left end of s in T by the least amount so that a prefix of the shifted pattern matches a suffix of s in T. If no such shift is possible, then shift P by n places to the right [6]. Following example depicts the rule. Consider the alignment of P and T given below.

```
          0                 1
          1 2 3  4 5  6 7  8  9 0 1 2 3 4 5
    T:    P I T X B  L C  A B V I X A S T
                          *
    P:       Q P A  B  F L  A  B
             1 2 3  4  5 6  7  8
```

When the mismatch occurs at position 6 of P and position 7 of T, s = AB and s′ occurs in P starting at position 3. Hence P is shifted right by four places resulting in the following alignment.

```
          0                 1
          1 2 3  4 5  6 7  8  9 0 1 2 3 4 5
    T:    P I T X B  L C  A B V I X A S T
    P:                    Q P  A B  F L A B
```

The good suffix rule is also modified to accommodate the variation that can happen in the pattern based on the PFM. In the pattern (Transcription Factor Binding Motif) the positions that may vary are represented with a symbol. For example the binding motif for TF NFIC::TLXI (fig1&2) is represented with 'TGGC*****GCCA*'. While checking for the suffix in the pattern '*' is considered as equal to every alphabet (A, T, C, G) and suffixes are checked and shift distance is calculated. In the above motif ** is considered as equal to A* and C** is considered as equal to CA*. And the shift distance is calculated on the basis of that.

Shift distance table for the good suffix rule of the pattern 'TGGC*****GCCA*' is given in the table 4

TABLE 4
GOOD SUFFIX RULE PRE PROCESSING FOR TF NFIC::TLXI

1	2	3	4	5	6	7	8	9	10	11	12	13	14
T	G	G	C	*	*	*	*	*	G	C	C	A	*
14	14	14	14	4	4	4	4	4	4	4	4	4	1

The value for the column 13 is 4 because the suffix pattern 'A*' is at a shift distance of 4 to left of the pattern because we consider 'A*' and '*G' as equal and '*G' is at position 9 and 10.

4.3 Algorithm

The idea of the Boyer–Moore algorithm is to call good suffix rule and bad character rule on mismatch. Both will return the shift distance and make the larger shift among that. Below shows the skeleton of the modified algorithm to match the position based vague pattern.

- Match the pattern (P-TF Binding motif) from right to left against the Text (T- DNA sequence).
- If mismatch occurs Check PFM for that position
  - o If value of mismatched alphabet in PFM>0
    - ▪ Consider it as a match and continue
  - o Else
    - ▪ call(bad character rule && good suffix rule)
    - ▪ Make the larger shift
- End

The good suffix rule and bad character rule is modified as mentioned earlier to accommodate uncertainty in the pattern. The proposed algorithm works correctly for pattern matches where position specific variation in the pattern is present. The algorithm scan the PFM only when the mismatch occurs and PFM look up takes only a constant time. So in overall it takes only a polynomial time complexity like the Boyer-Moore algorithm for pattern match.

5 Result

We implemented the proposed algorithm in java to find potential binding sites for the TF. A short synthetic DNA sequence and PFM for the TF was given as the input.
DNA-Sequence:
TCAAGGCACGTAGCTTAGCTATACGTAGCTTGACGACTGAT-
TAGCGCTATGCTATGCTAGTTGATGATCCAGGTTCTCTCGAGAGATC-
GATGCTAGCCTGCTATATAGAGAGACACCCCCAGAGATCGTATAGCCTCTA-
GAGCTAGCTGCGCTAGCGACGAGAGAGAGAGAGAGAGTATATAGACA-
GACTGCTGCATATGTACGATAGAAATGATTAGATTCAGTAAGAAC-
TAGGTCAAGGCCTGATCGCTATAGATACATAGCTCGGTGCGA-
TACGTCGTGACGCTGGCAT-
GACTCGTACGTCGCGACGTCTTGTCGTCGTCGCTCGTCACA-
TAGCTGTACCGTTCAAGTCGTGTCACATGCTGCTGCAAAAAAATGCACG-
TACCCCGTGTCGTCGCTGGATATATATAGCTCGGCGCCA-
CATGCTGCCATGCCACACAGTCACACA-
CACGTCGCTCGCAGTCGCAGTCGTTCTCGACACA-
TACGTCGCTGACGTCGCAGTCGCACATA-
TAGCGCTCGTCGCCTGGTGTCTCTCGGTGTGTGTGGAGACATACCTGAC-
TACGTACGATACTAGATGCTCCTTCTCTGATGACGATGCAGACTCGTAGATT-
CAATAACAGTACGTCGCTTACAGTCGCACTCGCTGCAGATCGTCA-
GACTCGTAGATTCGGTAACTATCTGTAGTAGTGTATAGAGAGAGA-
GACCCCCCCCCCCCCCTTCTAGGCTTTAAAAAGTGTGTGTGTTT.

TABLE 5
Input PFM

A	0	0	0	1	1	0	0	0
C	0	0	1	0	0	0	0	1
G	1	0	0	0	0	1	1	0
T	1	1	0	1	0	0	1	0

The fuzzy pattern specified by the PFM is '*TC*AG*C'. Implemented program returned the following result (Table 5).

TABLE 6
Input PFM

Pattern	Index
GTCAAGGC	238
TTCAAGTC	350
TTCTAGGC	726

The output is found to be 100% accurate.

6 Conclusion

A fast pattern searching algorithm that allows position based variation in the pattern is proposed. This algorithm can be effectively used for locating the potential Transcription Factor binding sites along the DNA. Similarly the proposed algorithm can be used for any application that allow position based variation in the pattern. The uncertainty along the position can be specified using a position based presence matrix like Position Frequency Matrix in the case of TFs. The search procedure can be coupled with any simple scoring function to effectively return the top N results, when the probability of occurring different symbols from Σ is different in the uncertain positions. The same technique can also be used along with Knuth Morris Pratt (KMP) algorithm for fixed pattern matching to match position based approximate patterns.

References

[1] Thomas H Corman, Charles E. Leiserson, Ronald L. Rivest & Clifford Stein "Introduction to AlgorithmsString matching", EEE Edition, 2nd Edition, Page no 906-907.

[2] A. Coller and L. Kruglyak. It's the sequence, stupid! Science, 322(5900):380–381, 2008.

[3] J. Galas and A. Schmitz. DNAase footprinting a simple method for the detection of protein-DNA binding specificity. Nucleic Acids Research, 5(9):3157–3170, 1978..

[4] G. Berg and P. H. von Hippel. Selection of DNA binding sites by regulatory proteins. Journal of Molecular Biology, 193(4):723–743, 1987.

[5] Tarhio, J., Ukkonen, E. (1993) Approximate Boyer-Moore String Matching. SIAM J. Comput., 22, pp. 243-260.

[6] Boyer, R.S., Moore, J.S. (1977) A fast string searching algorithm. Communications of the ACM, 10(20), pp. 762-772.

[7] G. Roider, A. Kanhere, T. Manke, and M. Vingron. Predicting transcription factor affinities to DNA from a biophysical model. Bioinformatics, 23:134–141, 2007..

[8] D. Schneider and M. Stephens. Sequence logos: a new way to display consensus sequences. Nucleic Acids Research, 18(20):6097–6100, 1990.

[9] Matys, O. V. Kel-Margoulis, E. Fricke, I. Liebich, S. Land, A. Barre-Dirrie, I. Reuter, D. Chekmenev, M. Krull, K. Hornischer, N. Voss, P. Stegmaier, B. Lewicki-Potapov, H. Saxel, A. E. Kel, and E. Wingender. TRANSFAC and its module TRANSCompel: transcriptional gene regulation in eukaryotes. Nucleic acids research, 34(Database issue):D108–D110, 2006.

[10] Mathelier, X. Zhao, A. W. Zhang, F. Parcy, R. Worsley-Hunt, D. J. Arenillas, S. Buchman, C.-y. Y. Chen, A. Chou, H. Ienasescu, J. Lim, C. Shyr, G. Tan, M. Zhou, B. Lenhard, A. Sandelin, and W. W. Wasserman. JASPAR 2014: an extensively expanded and updated open-access database of transcription factor binding profiles. Nucleic acids research, 42(Database issue):gkt997–D147, 2014.

[11] E. Newburger and M. L. Bulyk. UniPROBE: an online database of protein binding microarray data on protein-DNA interactions. Nucleic acids research, 37(Database issue):D77–D82, 2009.

[12] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

[13] Ricardo Baeza-Yates, Gonzalo Novarro, "Fast Approximate string matching in a Dictionary",Bulletin of the Technical Committee, 2000

[14] *Knuth, Donald; Morris, James H.; Pratt, Vaughan (1977).* "Fast pattern matching in strings". *SIAM Journal on Computing 6 (2): 323–350.* doi:10.1137/0206024.