Research Journal of Computer and Information Technology Sciences E-ISSN 2320 – 6527
Vol. **4(4),** 19-23, April **(2016)** *Res. J. Computer and IT Sci.*

Approximate Matching with Two Dimensional Contexts Free Grammars

Pawan Kumar Patnaik[1*], M.V. Padmavati[1] and Jyoti Singh[2]
[1]Deparatment of Computer Science and Engineering, BIT, Durg, CG, India
[2]Chhattisgarh Professional Examination Board, Raipur, CG, India
pawanpatnaik@yahoo.com

Available online at: www.isca.in, www.isca.me
Received 24th February 2016, revised 3rd April 2015, accepted 18th April 2016

Abstract

This paper introducesapproximation algorithm for matching a given text array approximately with text arrays generated by two dimensional Context Free Grammars. The CYK algorithm is extended for checking membership in 2D Context Free Languages. The algorithm outputs the minimal cost of finding such a match.

Keywords: Matching, Two Dimensional Contexts Free Grammars.

Introduction

Pattern matching has applications in the fields of pattern recognition, image processing, computer vision etc. In one dimension, this problem is referred to as string matching. String matching has got its applications in the fields of text editing, text searching, data base search, artificial intelligence, information retrieval etc. There are many instances in which one needs to find the occurrences of more than one user-defined pattern in the given text. This problem is known as multiple pattern matching. Library bibliographic search program is one such application. In two dimensions this problem is referred to as pattern matching. In many applications like computational biology, it is desirable to find the approximate matches of the pattern in the given text rather than the exact match.

An approximation algorithm is proposed in this paper for matching a given input image I with images generated by two dimensional Context Free Grammars. In the first scheme, the algorithm matches the given image I with images generated by 2D grammar approximately, without any row gaps and column gaps. The algorithm will output the minimal cost of finding such a match. Myers has designed approximate algorithm for determining the membership of a string of length *n* in L(G), where L(G) is a Context Free Language generated by the CFGG of size P[1].

Section 2 of this paper describes the background of the work presented. Section 3 deals edit distance algorithm with complexity analysis with example. In Section 4, a procedure for approximately matching in two dimensional Context Free Grammars has been discussed.

Preliminaries

Matrix Grammars: Matrix Grammars were studied in Abstract families of matrices and picture languages are a generation mechanism to generate rectangular arrays[2].

In this type of grammars first string is derived horizontally as intermediates and then the vertical columns of the array are derived. According to Siromoney G., Siromoney R. and Krithivasan K., all four types of grammars of Chomsky Hierarchy are considered in the horizontal direction and only type – 3 grammars are considered in the vertical direction and thus the four types of Matrix Grammars are called Regular Matrix Grammars (RMG), Context Free Matrix Grammars (CFMG), Context Sensitive Matrix Grammars (CSMG) and Phrase Structured Matrix Grammars (PSMG)[2].

Throughout our research work we have considered Context Free Grammars in the horizontal direction and vertical direction. We denote Matrix Grammars as (X:Y)MG where X, Y Є {CF, R}.

Definition 2.1.1: Let $\sum$ be an alphabet set – a finite non-empty set of symbols. A Matrix (or an image) over $\sum$ is an $m \times n$ rectangular array of symbols from $\sum$ where $m,n \geq 0$. The set of all matrices over $\sum$ (including ε) is denoted by $\sum^{**}$ and $\sum^{++} = \sum^{**} - \{\varepsilon\}$, where ε is the empty image.

Definition 2.1.2: R (I) and C (I) respectively represent the number of rows and columns of given matrix I.

Definition 2.1.3: Let $\sum^{*}$ denote the set of horizontal sequences of letters from $\sum$ and $\sum^{+}= \sum^{*}- \{\varepsilon\}$, where ε is the identity element (of length zero).$\sum^{*}$denotes the set of all vertical sequences of letters over$\sum$, and $\sum^{+}= \sum^{*} - \{\varepsilon\}$. Length of the given string s is denoted by |s|. Precisely, if s Є $\sum^{+}$then |s| = C(s) and if s Є $\sum^{+}$ then |s| = R(s).

For strings a and b, $a = a_1 \ldots\ldots\ldots a_n$, $b = b_1 \ldots\ldots\ldots\ldots b_m$, the concatenation (product) of a and b is defined as a.b and $a.b= a_1 \ldots\ldots\ldots a_n\ b_1 \ldots\ldots\ldots\ldots b_m$. Two types of concatenation i.e. row and column concatenations (row and column product) are defined in case of matrices.

Definition 2.1.4: We will use the operators Θ for row concatenation and ϕ for column concatenation. If

$$X = \begin{matrix} a_{11} & . & . & . & a_{1n} \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ a_{m1} & . & . & . & a_{mn} \end{matrix}$$

$$Y = \begin{matrix} b_{11} & . & . & . & b_{1n'} \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ b_{m'1} & . & . & . & b_{m'n'} \end{matrix}$$

XϕY is defined only when at least one of them is ε or m=m' and is given by

$$X\phi Y = \begin{matrix} a_{11} & . & . & a_{1n} & b_{11} & . & . & b_{1n'} \\ & . & . & . \; . & . & . & . & \\ & . & . & . & . & . & . & . \\ a_{m1} & . & . & a_{mn} & b_{m1} & . & . & b_{mn'} \end{matrix}$$

XΘY is defined only when atleastone of them is ε or n=n' and is given by

$$X\Theta Y = \begin{matrix} a_{11} & . & . & . & a_{1n} \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ a_{m1} & . & . & . & a_{mn} \\ \\ b_{11} & . & . & . & b_{1n} \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ b_{m'1} & . & . & . & b_{m'n} \end{matrix}$$

Definition 2.1.5: Let A be a matrix (or an image) defined over ∑then $(A)^{i+1} = (A)^i \phi$ x, and$(A)_{i+1} = (A)_i \Theta A$, $i \geq 1$.

Note: 2.1.1 to 2.1.5 are definitions according to Siromoney G., Siromoney R. and Krithivasan K.[2,3]. In this paper, we define mapping of a matrix (or an image) as follows. But both the definitions are equivalent.

Definition 2.1.6: Let us define a mapping χas follows: $\chi : \sum^+ \rightarrow \sum_+$.

For any string s =$a_1\, a_2 \ldots\ldots\ldots\ldots\ldots a_n \in \sum^+$

$$\chi(s) = \begin{matrix} a_1 \\ a_2 \\ . \\ . \\ . \\ a_n \end{matrix}$$

This means writing the string s vertically.
Formally, if s = $a_1\; a_2 \ldots\ldots\ldots\ldots\ldots a_n \in \sum^+$, $\chi(s) = a_1\Theta a_2\Theta\ldots\ldots\ldots\Theta a_n$.

Definition 2.1.7: Let us define a matrix (or an image) as follows: let $c_1,c_2,\ldots\ldots\ldots\ldots,c_n \in \sum^+$are strings of same length. I = $c_1\Theta c_2\Theta\ldots\ldots\Theta c_n$ is the matrix (or an image) represented by the image $\chi(c_1)\phi\chi(c_2)\phi\ldots\ldots\ldots\ldots\phi\chi(c_n)$.

Example: if c_1 = 1 2 3, c_2 = d e f , c_3 = \$ a c then I = $c_1\Theta c_2\Theta c_3 = \chi(c_1)\ \phi\chi(c_2)\phi\chi(c_3)$ is the image

1	d	\$
2	e	a
3	f	c

Edit Distance

Definition 2.2.1: Given two strings x = $x_1\, x_2\ldots\ldots\ldots\ldots.x_n$, y = $y_1\, y_2\ldots\ldots\ldots\ldots.y_n$ and a cost function δ, we define the distance between two strings xand y as
$\delta(x, y) = \sum_{i=1}^{n}\ {}_{=1}{}^{n}\delta(a_i,b_i)$
Where δ(a,b) is the cost of matching a with b. δ is called the matching cost function.

Definition 2.2.2: Given a CFG G = <N,T,P,S> and a string w Є T^* define
δ(G,w) = min({δ(w',w) | w' Є L(G)})
δ(G,w) is the least cost matching of w with G.

Definition 2.2.3: δ(M,I)= min({$\delta(I_1,I)$ | I_1 Є L(M)}) where M is the given matrix grammar and I is an image. δ(M,I) is the distance of I from M.

Note that $\delta(I_1,I_2)$ is the distance between the two images I_1 and I_2 and they are defined in the respective approximation schemes. Note: we have used δ for cost function in all the definitions. Depending on the parameter proper definition must be used.

Algorithm

A two-dimensional matrix, N[0..|x|,0..|y|] is used to hold the edit distance values:
N[i,j] ←d(x[1..i], y[1..j])
N[0,0] ←0
fori=1to |x| do, N[i,0] ←i
for j:=1 to |y| do, N[0,j]← j
fori=1to |x| do
for j=1 to |y| do
ifx[i]=y[j] then k←0 else k←1
N[i,j]←min(N[i-1,j-1], N[i-1, j] + 1,N[i, j-1] + 1)

Analysis: O (|x|*|y|) is the time-complexity of the algorithm and is O(n^2) if the lengths of both strings is about 'n'. O (|x|*|y|) is also the space-complexity if the whole of the matrix is kept for a trace-back to find an optimal alignment.

Example: To find Edit Distance between two string x= a b aandy = a b a.

Table-1
Output of Edit Distance algorithm

	j→	0	1	2	3
i↓			a	b	a
0		0	1	2	3
1	a	1	0	1	2
2	b	2	1	0	1
3	a	3	2	1	0

Design of Approximately Matching in two dimensional Context Free Grammar

4.1 Algorithm to find δ(G,w)
Input: given a CFG G = <N,T,P,S>in CNF, a string w = a_1 a_2 ………………a_n and a matching cost function δ.

Output: δ(G,w) and anyone of the strings which approximately matches with w. In the following algorithm, cost function is represented using C.

Algorithm:
C(X,i,j) ← ∞, ∀ 1 ≤ i, j ≤ n, ∀X Є N
ST(X,i,j) ← ϕ, ∀X Є N

For i = 1 to n do
∀X → a Є P
temp ← δ(a,a_i)
if (temp < C(X,i,i))
C(X,i,i) ← temp
ST(X,i,i) ← a

For l = 2 to n do
For i = 1 to n-l+1 do
j ← i + l– 1
For k = i to j-1 do
∀X → BDЄ P
temp ← C (B,i,k) + C(D,k+1,j)
if(temp < C(X,i,j))
C(X,i,j) ← temp
ST(X,i,j) ← ST(B,i,k) . ST(D,k+1,j)

Return C(S,1,n) and ST(S,1,n).

Also note that δ(G,w)= C(S,1,n) and S(G,w) = ST(S,1,n) where S(G,w) represents one of the strings which approximately matched by G with given input string.

The correctness of the algorithm can be proved from the correctness of the CYK algorithm.

Example: Consider the CFG
S → XY ǀ YZ
X→ YX ǀ a
Y → ZZ ǀ b
Z → XY ǀ a
And the input string is x = baab.

Table-2
Output of Edit Distance Algorithm

	j→	0	1	2	3	4
i↓			b	a	a	b
0		0	1	2	3	4
1	b	1	0	1	2	3
2	a	2	1	0	1	2
3	a	3	2	1	0	1
4	b	4	3	2	1	0

Input String: baab
Output is δ(G,w) and anyone of the strings which approximately matches with w are shown in Table-3.

Table-3
Output of approximately matching in two dimensional CFG

j→	1	2	3	4
i↓	b	a	a	b
1	b	b.a	-	-
2	-	a	a.a	a.a.b
3	-	-	a	a.b
4	-	-	-	b

Approximation Matching: Scheme 1: In this section we consider the first approximation scheme in which we match the given input image approximately with an image of same size generated by a two dimensional grammar.

Definition 4.2.1: Scheme 1
Given two images I_1 = $c_1 \Theta c_2 \Theta ....\Theta c_n$ and I_2 = $d_1 \Theta d_2 \Theta ....\Theta d_n$ of same size, we define the distance between I_1 and I_2 as
$\delta(I_1,I_2) = \sum_{i=1}^{n} \delta(c_i,d_i)$
If I_1 and I_2 are not of same size then $\delta(I_1,I_2) = \infty$.
Algorithm to find δ(M,I) as per scheme-1.
Input:

A (CF: CF) 2D grammar or MG M = <G,G'> where G = <N,T,P,S>be a CFG in CNF T = {A_1, A_2, ……….A_k} , G' = {G_1,G_2,……….G_k} where each G_i is a CFG in CNF corresponding to A_i.

An image $I = c_1\Theta c_2\Theta.........\Theta c_n$
Output: To find δ(M,I) as per scheme 1 and also one of the images to which the given input image is approximately matched.

Algorithm:
$W(S_x,i,j) \leftarrow \infty$, $1 \le i, j \le n$, $\forall S_x \in N$
$SW(S_x,i,j) \leftarrow \phi$, $1 \le i, j \le n$, $\forall S_x \in N$

Find $V^i_{A \times \in T}$, $S^i_{A \times \in T}$, $1 \le i \le n$ using the algorithm 4.1
$V^i_{Ax \in T} \leftarrow \delta(G_x, c_i)$
$S^i_{Ax \in T} \leftarrow S(G_x, c_i)$

For i = 1 to n do
$\forall S_x \rightarrow A \in P$
temp $\leftarrow V_A^i$
If (temp< $W(S_x,i,i)$)
$W(S_x,i,i) \leftarrow$ temp
$SW(S_x,i,i) \leftarrow \chi(S_A^i)$

For len = 2 to n do
For i = 1 to n-l+1 do
$j \leftarrow i + l - 1$
For k = i to j-1 do
$\forall S_x \rightarrow S_y S_z \in P$
temp $\leftarrow W(S_y,i,k) + W(S_z,k+1,j)$
if(temp < $W(S_x,i,j)$)
$W(S_x,i,j) \leftarrow$ temp
$SW(S_x,i,j) \leftarrow SW(S_y,i,k)\ \phi\ SW(S_z,k+1,j)$

Return W(S,1,n) and SW(S,1,n).

$W(S_x,i,j)$ denotes the least cost of matching of $c_i\Theta.........\Theta c_j$ approximately with an image generated by S_x. $SW(S_x,i,j)$ denotes one of the least cost images which approximately matched with the given input image $c_i\Theta.........\Theta c_j$. The above algorithm runs in $O(n^4 |P|)$ time, where the given input image is assumed to be of size n×n. The proof of correctness of this algorithm is a direct from that of set – CYK algorithm. In addition to that we are also reporting one of the images to which the given input image is approximately matched.

Example: The following Chomsky Normal Form two dimensional Grammar G defines the set of Images such that each column is Palindrome:
$E \rightarrow W\phi E \mid X_1 \Theta X_2 \mid Y_1 \Theta Y_2 \mid x \mid y$
$W \rightarrow X_1 \Theta X_2 \mid Y_1 \Theta Y_2 \mid x \mid y$
$X_2 \rightarrow W \Theta X_1 \mid x$
$Y_2 \rightarrow W \Theta Y_1 \mid y$
$X_1 \rightarrow x$
$Y_1 \rightarrow y$
if $c_1 = x\ y\ x$, $c_2 = y\ y\ y$ then $I = c_1\Theta c_2 = \chi(c_1)\phi\chi(c_2)$ is the image

$$\begin{matrix} x & y \\ y & y \\ x & y \end{matrix}$$

Table-4
Output of Edit Distance Algorithm

	j →	0	1	2	3
i ↓			x	y	x
0		0	1	2	3
1	x	1	0	1	2
2	y	2	1	0	1
3	x	3	2	1	0

	1	2	3
	y	y	y
1	y	y y	y y y
2	-	y	y y
3	-	-	y

Input String:

$$I = \begin{matrix} x & y \\ y & y \\ x & y \end{matrix}$$

Output is δ(M,I) as per scheme-1 and also one of the images to which the given input image is approximately matched are shown in Table-5

Table-5
Output of approximately matching in two dimensional CFG

	1	2	3
	x	y	x
1	x	x y	x y x
2	-	y	y x
3	-	-	x

	j →	0	1	2	3
i ↓			y	y	y
0		0	1	2	3
1	y	1	0	1	2
2	y	2	1	0	1
3	y	3	2	1	0

Conclusion

In this paper, we have proposed a scheme for approximately matching two dimensional Context Free Grammar. In this scheme, we have proposed algorithm that runs in O (n^4 |P|) time for (CF: CF) 2D Grammar by modifying the algorithm proposed by Myers [1].

References

1. Myers G. (1995). Approximately matching context-free languages. *Information Processing Letters*, 54(2), 85-92.

2. Siromoney G., Siromoney R. and Krithivasan K. (1972). Abstract families of matrices and picture languages. *Computer Graphics and Image Processing*, 1(3), 284-307.

3. Siromoney G., Siromoney R. and Krithivasan K. (1973). Picture languages with array rewriting rules. *Information and Control*, 22(5), 447-470,

4. Subramanian K.G., Pan L., Lee S.K. and Nagar A.K. (2009). P Systems and Context-free 2D Picture Languages. Fourth International Conference on Bio-Inspired Computing IEE, 1-5.

5. ReghizziS. C. and Pradella M. (2008). A CKY parser for picture grammars. *Information Processing Letters,* 105, 213-217.