ASPECTOS SEQUENCIAIS DA DECODIFICAÇÃO SINTÁTICA

FLÁVIO ROBERTO DIAS VELASCO

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIA (M. Sc.).

Aprovada por:

Presidente

RIO DE JANEIRO

ESTADO DA GUNABARA - BRASIL

JANEIRO DE 1973

À Ana

AGRADECIMENTOS

Ao professor Celso de Renna e Souza pela sugestão, orientação e constante assistência a este trabalho.

A Tânia, minha irmã, por ter datilografado esta tese.

RESUMO

O presente trabalho procura usar a informação sintática contida nas sentenças pertencentes a uma linguagem formal na sua decodificação quando as mesmas são transmitidas através um canal ruidoso. É adaptado o algoritmo de Fano da decodificação sequencial na decodificação sintática. O algoritmo é simulado em computador IBM 370 para três diferentes linguagens e canais de transmissão.

ABSTRACT

The present work uses the syntactical information of the sentences of a formal language in their decoding, after they have passed through a noisy channel. Fano's algorithm for sequential decoding is adapted to syntactical decoding. The algorithm is simulated in the IBM 370 for three diferent sources and channels. The results of the simulation are discussed and analysed.

ÍNDICE

CAPÍTULO I

INTRODUÇÃO

Um "sistema de comunicação" é tradicionalmente repre - sentado por um diagrama de blocos como o da figura 1.1.

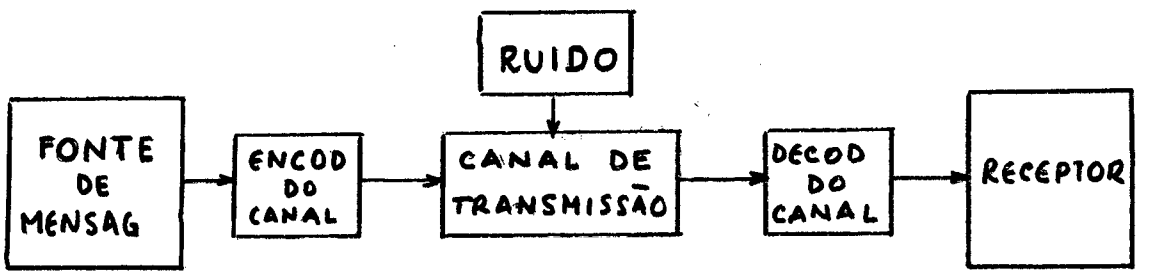


figura 1.1

A "fonte de mensagens" ou simplesmente "fonte" é a pessoa ou máquina responsável pela informação que deve ser comunicada. O "encodificador do canal" associa a cada mensagem produzida pela fonte uma mensagem capaz de ser transmitida pelo canal de transmissão. O "canal de transmissão" é o meio pelo qual a mensagem é transmitida. O "ruído" representa as perturbações do meio que afetam a mensagem. O "decodificador do canal" procura reproduzir a mensagem gerada pela fonte a partir da recebida na saída do canal. Finalmente, o "receptor" é a pessoa ou máquina a que se destina a informação.

A Teoria da Informação é uma tentativa de construir modelos matemáticos para cada um dos blocos da figura 1.1.

Se considerarmos fontes cujas mensagens estejam escritas em alguma linguagem natural, como português ou inglês, o modelo da Teoria da Informação para este tipo de fonte é o modelo

markoviano. Sequências produzidas por uma fonte markoviana adequada em vários trechos se parecem com trechos de senten - ças de uma linguagem natural. Contudo, a aproximação markoviana de uma linguagem natural revela-se insuficiente quando exposta a um exame mais rigoroso. Embora algumas das sequências da fonte markoviana se pareçam com sequências de linguagem natural, na realidade essas sequências não são corretas sintaticamente, portanto não pertencem à esta linguagem natural.

Isto nos leva à crença de que uma linguagem natural não pode ser descrita de um modo exclusivamente probalístico.

Um modelo mais completo para uma linguagem produzida por uma "fonte natural" é o que inclue além dos aspectos léxicos da linguagem, os aspectos sintáticos e semânticos. Isto se baseia no fato que numa mensagem produzida por uma fonte natural podemos distinguir três níveis de informação.

O primeiro nível é o da informação léxica, ou seja, a informação contida nos elementos ou "palavras" da mensagem isoladamente. O segundo nível é o da informação sintática contida na ordem das palavras e nas suas intercorrelações. Final - mente temos a informação semântica, responsável pelo sentido da mensagem, o que a frase "quer dizer". Esta situação pode ser visualizada através do diagrama de blocos da figura 1.2.

O modelo que usaremos restringir-se-á aos aspectos sintáticos e léxicos da informação, pois as ferramentas necessárias para a manipulação da informação semântica ainda estão em desenvolvimento.

Na figura 1.3 temos o diagrama de blocos do sistema de comunicação, onde o receptor é suposto análogo à fonte. Além disso o "canal" da figura 1.3 inclue tanto o ruído como encodificador e decodificador do canal.

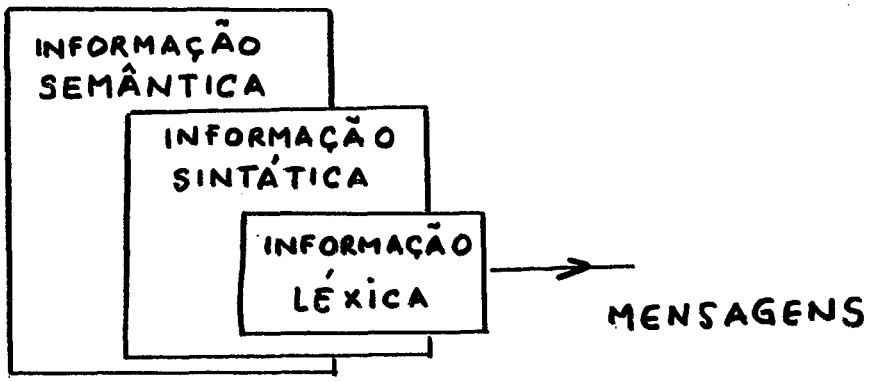


figura 1.2

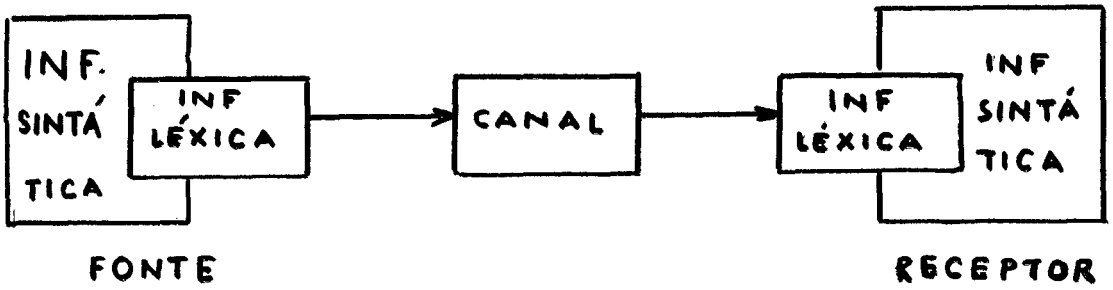


figura 1.3

A descrição sintática das linguagens naturais que consideraremos no presente trabalho é devida a Chomsky (,). Segundo Chomsky, uma linguagem é sintaticamente descrita através uma gramática formal pelas "regras de reescrita".

O presente trabalho preocupa-se com o "lado direito" do modelo descrito pela figura 1.3, ou seja o receptor. Procuraremos aproveitar a informação sintática contida nas mensagens para sua decodificação, após as mensagens terem passado pelo

canal de transmissão.

De um certo modo, a sintaxe de uma linguagem natural pode ser encarada como uma maneira de se introduzir redundância para se proteger a informação contida na mensagem. As linguagens naturais são pois "códigos naturais" para a transmissão de informação.

Podemos constatar esta redundância no fato que o conjunto de frases de uma linguagem é um subconjunto do conjunto de todas as possíveis concatenações de palavras do vocabulário. Esta redundância é que procuraremos explorar para a correção de êrros eventualmente introduzidos pelo canal de transmissão.

A idéia de se usar a sintaxe das linguagens para ajudar a decodificação é recente e pouca coisa se tem escrito sobre o assunto. Tentativas esporádicas de ligar a Teoria da Informação com a teoria das linguagens formais foram feitas por Banerjii (3) e Conant (20). A idéia de decodificação sintática é devida a Souza (24 ,25). Em (24) Souza propôs o presente modelo para fontes sintáticas de informação, estabeleceu propriedades gerais desejáveis para os decodificadores sintáticos, além de construir um algoritmo decodificador "mínima distância". Dierks (10) desenvolveu um procedimento para se determinar a "separação mínima" de uma linguagem regular, que se usada como ferramenta de análise diz se é possível ter um único "parse" de uma sequência com "erros".

Dierks formulou também condições para existência de "corretor" de uma linguagem regular onde se introduziu "erros".

Neste estudo procurou-se aplicar os princípios da decodificação sequencial, usada na Teoria da Informação para os códigos convolucionais, na decodificação sintática.

É feita a adaptação do algoritmo de Fano para a decodi-

ficação sintática, e o mesmo é simulado em computador na decodificação de vários tipos de linguagens e diversas espécies de canal de transmissão.

As demonstrações de alguns teoremas serão omitidas quando as mesmas não forem essenciais para a compreenção do texto e já existirem na literatura. Assim é que as demonstrações dos teoremas 2.1 e 2.2 podem ser encontrados em Hopcroft e Ulman (16) e o teorema 2.3 em Souza (25).

A organização do trabalho é a seguinte: no capítulo II é feito um breve estudo das linguagens formais e de alguns modelos de gramáticas programadas. No capítulo III é estudado o canal de transmissão e os esquemas de decodificação. A decodificação sintática sequencial é estudada no capítulo IV e os resultados da simulação são apresentados e analisados no capítulo V. No capítulo VI expôs-se as conclusões e indica-se alguns rumos para posteriores pesquisas.

CAPÍTULO II

LINGUAGENS FORMAIS E GRAMÁTICAS PROGRAMADAS

2.1 Gramáticas Formais

Dado um conjunto A, definimos A.A ($=A^2$) como o conjunto de todas as concatenações possíveis de elementos de A com elementos de A.

Exemplo 2.1 seja $A = \{a,b\}$
então $A^2 = \{aa,ab,ba,bb\}$

A palavra vazia, Λ é definida como sendo a sequência que não contém símbolo algum.

A^* é definido como:

$$A^* = \{\Lambda\} \cup A \cup A^2 \cup A^3 \cup A^4 \cup \ldots \quad \text{e}$$
$$A^+ = A^* - \{\Lambda\} = A \cup A^2 \cup A^3 \cup \ldots$$

Definição 2.1 Uma gramática G é uma quádrupla $G = \langle N,T,P,S \rangle$, onde:

1. N é um conunto finito não vazio chamado "vocabulário não terminal". Os elementos de N são os "não terminais" ou "variáveis".
2. T é um conjunto finito não vazio chamado "vocabulário terminal". Os elementos de T são os "terminais".
3. $S \in N$, S é um não terminal especial.
4. $N \cap T = \emptyset$, $N \cup T = V$

5. P é um conjunto finito não vazio de regras da forma $x \to y$ (lê-se x é reescrito como y), onde $x \in V^+$ e $y \in V^*$. Os elementos de P são chamados de "produções" ou "regras de reescrita".

Para uma gramática $G = \langle N,T,P,S \rangle$ e $x,y \in V^*$, dizemos que y é "diretamente derivável" de x, $x \Rightarrow y$ se existe $r,s,u_1,u_2 \in V^*$ tal que $x = u_1 r u_2$ e $y = u_1 s u_2$ e $r \to s \in P$.

Para $x,y \in V^*$, y é "derivável" de x, $x \overset{*}{\Rightarrow} y$ se $x = y$ ou existe uma cadeia $x = u_0, u_1, u_2, \ldots u_k = y$, tal que $u_{i-1} \Rightarrow u_i$ $\forall$ $1 \leq i \leq k$.

Exemplo 2.2 seja a gramática $G = \langle N,T,P,S \rangle$, onde:
$N = \{S\}$, $T = \{0,1,C\}$ e $P = \{S \to 0S0,\ S \to 1S1,\ S \to C\}$
seja $r = 010S010$ e $v = 0101S1010$, então $r \Rightarrow v$.
seja também $s = 01S10$ e $t = 0101C1010$, então $s \overset{*}{\Rightarrow} t$ pois existe a cadeia:
$u_0 = s$, $u_1 = r$, $u_2 = v$, $u_3 = t$, tal que
$u_{i-1} \Rightarrow u_i$, $1 \leq i \leq 3$.

Definimos "comprimento" de uma sequência x, $|x|$ como o número de símbolos de x. $|\Lambda|$ é por definição igual a zero.

Definição 2.2 Uma gramática G é sensível ao contexto, ou do tipo 1 se toda produção for da forma: $u_1 A u_2 \to u_1 x u_2$, onde $u_1,u_2 \in V^*$ e $A \in N, x \in V^*$.

Definição 2.3 Uma gramática G é livre de contexto, ou do tipo 2 se toda produção for da forma $A \to x$, onde $A \in N$ e $x \in V^*$.

Definição 2.4 Uma gramática G é dita regular ou do tipo 3 se toda a produção for do tipo $A \to aB$ ou $A \to a$, onde $a \in T$ e $A, B \in N$.

As gramáticas do tipo 0 são aquelas às quais não fazemos restrições à forma de suas produções.

Definição 2.5 A "linguagem" L(G) gerada pela gramática $G=\langle N,T,P,S\rangle$ é: $L(G)=\{x \mid x \in T^* \text{ e } S \overset{*}{\Rightarrow} x\}$.

Duas gramáticas são ditas "equivalentes" se as linguagens geradas por estas gramáticas forem iguais, ou seja, a gramática G é equivalente a G' sse L(G)=L(G').

Definição 2.6 Uma "forma sentencial" x de uma gramática G é uma sequência pertencente a V* tal que $S \overset{*}{\Rightarrow} x$.

Uma linguagem L é dita ser regular, livre de contexto ou sensível ao contexto se existe uma gramática G tal que L=L(G) e G seja respectivamente regular, livre de contexto e sensível ao contexto.

2.2 Gramáticas livres de contexto

As gramáticas livres de contexto são, como vimos, aquelas nas quais o lado esquerdo das produções é composto por símbolos pertencentes ao vocabulário não terminal.

Toda derivação de uma gramática livre de contexto pode ser descrita através uma árvore, a "árvore de derivação".

Exemplo 2.3 Seja a gramática livre de contexto $G=\langle N,T,P,S\rangle$ onde $N=\{A,B,S\}$, $T=\{a,b\}$ e P dado pelas seguintes produções:

$S \to aA$	$A \to b$
$S \to bB$	$B \to BBb$
$A \to AaA$	$B \to aS$
$A \to Sb$	$B \to a$

Consideremos a derivação em G:

$S \Rightarrow aA \Rightarrow aAaA \Rightarrow abaA \Rightarrow abaSb \Rightarrow ababBb \Rightarrow ababab$

A árvore de derivação para a derivação acima é:

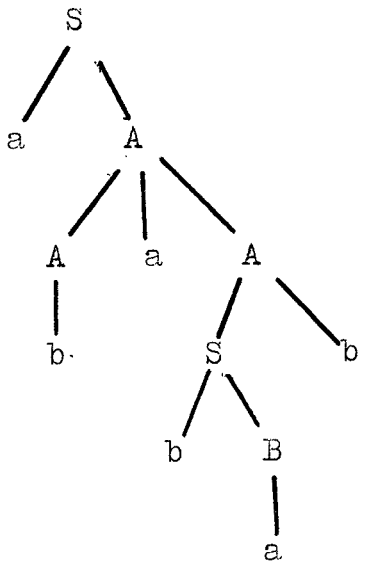


figura 2.1

O procedimento inverso ao da derivação e que consiste em dada uma sequência pertencente a uma linguagem achar uma deri - vação que possa gerar esta sequência é chamado "parsing" da sequência.

Na derivação do exemplo 2.3 escolhemos a cada passo da derivação o não terminal mais a esquerda para ser reescrito. Uma derivação em que isto acontece é chamada "derivação pela esquerda" (leftmost derivation).

Teorema 2.1 Dada uma gramática livre de contexto G, se $x \in L(G)$ então existe uma derivação pela esquerda ($S \overset{*}{\underset{L}{\Rightarrow}} x$), que produz x.

Ou seja, se dada uma gramática livre de contexto limitarmos as derivações a derivação pela esquerda, a linguagem gerada pela gramática será a mesma. A imposição de derivação pela esquerda não altera o poder gerador da gramática.

Definição 2.7 Uma gramática $G=\langle N,T,P,S\rangle$ é dita estar na "forma reduzida" sse $\forall A \in N$ existe $x \in T^*$ tal que $A \overset{*}{\Rightarrow} x$.

Teorema 2.2 Para toda gramática livre de contexto existe uma gramática equivalente G' na forma reduzida.

O que fazemos em G para obter G' é simplesmente "expurgar" N dos não terminais "estéreis" e as produções que contenham estes não terminais. Prova-se que este procedimento não altera a linguagem gerada pela gramática. Um não terminal A é "estéril" se o conjunto $\{ x \mid x \in T^* \text{ e } A \overset{*}{\Rightarrow} x \} = \emptyset$.

2.3 Gramáticas programadas

Até agora consideramos as gramáticas formais como descrições da estrutura das linguagens e não como mecanismos capazes de gerar as sentenças dessas linguagens.

A maneira natural de se fazer uma gramática gerar sentenças é atribuir probabilidades às produções. Há na literatura diversos enfoques a este problema.

O modelo mais geral é que faz a probabilidade de se aplicar uma produção numa derivação depender de todas as produções já aplicadas. Uma simplificação a esta idéia é proposta por Saloma (22), em cujas "gramáticas probabilísticas" a probabilidade de se aplicar uma produção só depende da última produção aplicada.

Uma "gramática probabilítica" é uma gramática na qual é associada à produção i um vetor estocástico f_i de dimenção igual ao número de produções da gramática. O j-ésimo componente deste vetor indica a probabilidade de que a j-ésima produção seja aplicada após a aplicação da i-ésima produção. Um vetor f_0 dá a distribuição de probabilidades em que a primeira produção deve ser escolhida. Deste modo a cada forma sentencial da gramática é associada uma probabilidade e em particular a cada sequência da linguagem.

Dentre os vários modelos existentes escolhemos as "gramáticas programadas" ($_{21}$,$_{25}$) como modelo da fonte por sua generalidade e por sua facilidade de implementação.

No nosso estudo nos limitaremos às gramáticas programadas com produções livres de contexto.

Definição 2.8 Uma "gramática programada livre de contexto" G, abreviadamente gplc, é a sêxtupla $G=\langle N,T,P,S,J,F\rangle$, onde $G'=\langle N,T,P,S\rangle$ é uma gramárica livre de contexto sem produções $-\Lambda$ (produções cujo lado direito é a palavra vazia). J é um conjunto de rótulo de produções e F é uma coleção de campos associados às produções de P.

Cada produção de G terá a seguinte forma:

i) $A_i \to^{\psi_i} S(V_i)\ F(W_i)$, onde $i \in J, V_i, W_i \subset J$.

V_i indica as produções que podem ser aplicadas após uma aplicação bem sucedida de i. Se i falha, isto é, se não há nenhum A_i na forma sentencial intermediária, a próxima produção a ser usada deve ser escolhida no campo de fracasso W_i. Aplica-se sempre a produção ao terminal A_i mais à esquerda.

É fácil ver que podemos fazer as gplc gerarem todas as linguagens livres de contexto, exceto $\{\Lambda\}$, simplesmente fazendo $V_i=W_i=J$. Podemos inclusive gerar linguagens sensíveis ao contexto com gramáticas programadas com produções livres de contexto.

Exemplo 2.4 Seja o seguinte exemplo de gplc:

$G=\langle N,T,P,S,J,F\rangle$, onde $N=\{S,A\}$, $T=\{0,1\}$, $J=\{1,2,3,4,5\}$ e produções da forma:

1)	S → 0AA	(2,4,5)	(1,2)
2)	S → 1S	(4,5)	(2,3)
3)	S → 0	(2,3,4)	(1,5)
4)	A → 1A	(3,4)	(2)
5)	A → 0	(3)	(2,5)

Definição 2.9 Uma "gramática programada livre de contexto com probabilidades", gplcp, é uma gplc $G=\langle N,T,P,S,J,F\rangle$ com distribuição de probabilidade associada a cada campo de sucesso e fracasso.

A forma típica de uma produção de uma gplcp é:

i) $A_i \rightarrow \Psi_i\ S(V_i)\ P(V_i)\ F(W_i)\ P(W_i)$, onde $V_i=\{j,k,l,\ldots\}\subset J$ e $P(V_i)=(Pr_i^S(j), Pr_i^S(k),..)$ onde $Pr_i^S(j)$ é a probabilidade de j ser escolhida após a aplicação bem sucedida da produção i.

Temos ainda que:

$$Pr_i^S(j)+Pr_i^S(k)+\ldots = {}_{j}\sum_{V_i} Pr_i^S(j)= 1.$$

Analogamente, $P(W_i)$ será composto de $Pr_i^F(j)$, onde $j \in W_i$.

As gplcp são inicializadas probabilisticamente por um vetor n-dimencional, onde n=#J. Uma vez inicializada, a gramática gera uma sentença pertencente à linguagem gerada pela gplc, com uma probabilidade associada a esta sentença.

As gplcp são mais gerais que as gramáticas probabilísticas no fato que a probabilidade de se aplicar uma produção depende das produções já usadas e não sòmente da última produção usada.

Várias simplificações interessantes podem ser obtidas ao impormos derivação pela esquerda às gramáticas programadas. Em particular, para gramáticas livres de contexto esta condição não altera seu poder, como vimos pelo teorema 2.2.

Teorema 2.3 Qualquer gplcp com derivação pela esquerda pode ser colocada numa forma equivalente sem campos de fracasso, bastando saber em alguns pontos da derivação um terminal à frente do que está sendo processado ( ).

Ao impormos derivação pela esquerda, o próximo não terminal a ser processado está de um modo geral do lado direito da produção que se está processando. A única exceção acontece quando o lado direito da produção considerada é composto só de terminais.

O procedimento de se eliminar os campos de fracasso é sobretudo adequado quando sabemos que a linguagem considerada é livre de contexto. Neste caso podemos assumir que as derivações são pela esquerda e colocar as produções na forma geral de Greibach.

Definição 2.10 Uma "gramática programada na forma padrão" ou simplesmente "gramática padrão" é uma gplcp sem campos de fracasso cujas produções estão na forma normal de Greibach e são da seguinte forma:
j) $A_j \rightarrow a_j Y_j^1 \ldots Y_j^n \quad (V_j)\ P(V_j)$, onde:
$a_j \in T$, $A_j \in N$, $Y_j^1 \ldots Y_j^n \in N^+$, e V_j contém sòmente rótulos de produções - Y_j';
ou j) $A_j \rightarrow a_j\ A{:}\ (V_j^A)\ P(V_j^A)\ B{:}\ (V_j^B)\ P(V_j^B) \ldots$
onde $A_j, A, B, \ldots \in N, a_j \in T$ e V_j^Y contém sòmente rótulos de produções - Y.

Para que a gramática padrão possa efetivamente gerar sentenças, associamos a ela um "vetor inicializante" n-dimensional, onde n é igual ao número de produções da gramática. Este vetor nos dará as probabilidades de se aplicar uma produção no primeiro passo da derivação. O vetor inicializante deverá ter nulas as coordenadas correspondentes às produções que não sejam produções - S.

Exemplo 2.5 Seja a seguinte gramática na forma padrão:
1) $S \rightarrow 0SS_0 \quad (1,2,3) \quad (0.5,0.25,0.25)$

2) $S \rightarrow 1SS_1$ (1,2,3) (0.5,0.25,0.25)

3) $S \rightarrow C$ S_0: (4) (1.0) S_1: (5) (1.0)

4) $S_0 \rightarrow 0$ S_0: (4) (1.0) S_1: (5) (1.0)

5) $S_1 \rightarrow 1$ S_0: (4) (1.0) S_1: (5) (1.0)

com o vetor inicializante:

(0.4,0.4,0.2,0,0)

Definição 2.11 Seja a gramática padrão $G = \langle N,T,P,S,J,F \rangle$ e $x,y \in V^*$ dizemos que y é diretamente p-derivável de x segundo G, $x \Rightarrow y$, sse:

1) $S \overset{*}{\underset{G'}{\Rightarrow}} x$, x é derivável de S por uma derivação pela esquerda em G', onde $G' = \langle N,T,P,S \rangle$.

2) $x \underset{G'}{\Rightarrow} y$ por uma derivação pela esquerda.

3) sendo j o rótulo da produção usada no item 2, e i o rótulo da última produção usada na derivação $S \overset{*}{\underset{G'}{\Rightarrow}} x$, então:

a) se a produçao i é da forma:

i) $A_i \rightarrow a_i A: (V_i^A)\ P(V_i^A)\ \ B: (V_i^B)\ P(V_i^B)\ldots,$

então $j \in V_i^Y$ onde Y é o não terminal do lado esquerdo da produção j.

b) se a produção i é da forma:

i) $A_i \rightarrow w_i\ \ (V_i)\ P(V_i)$, então $j \in V_i$

4) $S \Rightarrow_p x$, $x \in V^*$ se $S \rightarrow x \in P$, e a coordenada referente à produção $S \rightarrow x$ do vetor inicializante é diferente de zero.

Definimos analogamente $\overset{*}{\Rightarrow}_p$ como o fechamento transitivo da relação $\Rightarrow_p$.

Definição 2.12 L é a linguagem gerada pela gramática padrão G se $L = L_p(G) = \{ x \mid x \in T^* \text{ e } S \overset{*}{\Rightarrow}_p x \}$

Dada uma gramática na forma padrão G com vetor inicializante v_0, a cada sequência x pertencente a $L_p(G)$ está associada uma probabilidade de geração p(x) dada pela equação 2.1. Seja $l = l^1 l^2 \ldots l^n$ a sequencia de produções usadas na derivação de x.

$$p(x) = p_0(l^1)p(l^2/l^1)p(l^3/l^1l^2) \ldots p(l^n/l^1l^2 \ldots l^{n-1}) \qquad (2.1)$$

onde $p_0(l^1)$ é a probabilidade de que a produção l^1 seja usada no primeiro passo da derivação de x e é dada pelo vetor v_0.

CAPÍTULO III

CANAL DE TRANSMISSÃO E ESQUEMAS DE DECODIFICAÇÃO

3.1 Canal de transmissão

O segundo bloco do modelo do sistema de transmissão é o canal cujo modelo examinaremos a seguir.

Seja $A=\{a_1,a_2,\ldots a_n\}$ o conjunto de símbolos que desejamos transmitir, "símbolos de entrada" do canal. Um canal sem memória é o canal que associa a cada símbolo do conjunto A um outro símbolo do conjunto $B=\{b_1,b_2,\ldots b_m\}$ de "símbolos de saída" do canal, segundo probabilidades de transição $p(b_j/a_i)$, tal que $\sum_{j=1}^{m} p(b_j/a_i)=1$, como mostrado na figura 3.1. Assim é que se na entrada do canal "enviarmos" uma sequência $x \in A^*$, teremos na saída uma sequência $y \in B^*$, tal que $|x| = |y|$.

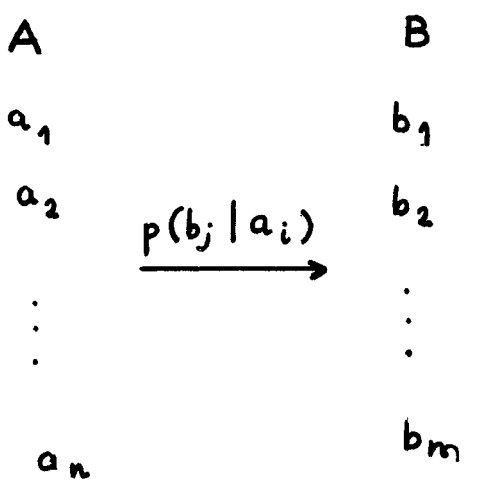


figura 3.1

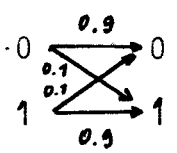
Um canal sem memória é completamente descrito por uma matriz n × m, onde n=#A e m=#B, chamada "matriz de transição", cujos elementos p_{ji} são as probabilidades de transição $p(b_j/a_i)$.

$$[p_{ij}] = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1m} \\ p_{21} & & \cdots & p_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ p_{n1} & & \cdots & p_{nm} \end{bmatrix}$$

Um "canal simétrico" é um canal sem memória tal que A=B e $\forall$ i, j, k, l, $p_{ij} = p_{jj}$ e $p_{ij} = p_{kl}$ se $k \neq l$ e $i \neq j$.

Exemplo 3.1 exemplo de canal simétrico:

Seja A = B = 0,1



matriz de transição:

$$\begin{bmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{bmatrix}$$

As sequências de símbolos produzidas pela fonte e que desejamos transmitir, formam um conjunto $X \subset A^*$. No caso da fonte ser "gramatical" temos que A = T e $X = L_p(G)$, onde para todo $x \in X$ existe uma probabilidade de geração p(x).

Definimos o conjunto Y de "sequências recebíveis" como o conjunto $Y = \{ y \mid y \in B^* \text{ e } \exists\, x \in X \text{ tal que } p(y/x) > 0 \}$. Ou seja, o conjunto de sequências que podemos receber enviando sequências de X através do canal de transmissão.

No caso de um canal sem memória, onde, $x = a_1 a_2 \ldots a_n$

e $y = b_1 b_2 \ldots b_n$,

$$p(y/x) = p(b_1/a_1)p(b_2/a_2) \ldots p(b_n/a_n) = \prod_{i=1}^{n} p(b_i/a_i), \qquad (3.1)$$

onde, $|x| = |y| = n$.

Se $|y| \neq |x|$ teremos que $p(y/x) = 0$.

3.2 Esquemas de decodificação

O problema de decodificação é tentar determinar qual a sequência mais provável de ter sido transmitida dada uma mensagem recebida.

Um procedimento que associa a cada mensagem recebida uma mensagem transmitida é chamado "esquema de decodificação".

Um esquema de decodificação induz uma partição $\tilde{\pi}$ no conjunto Y de sequências recebíveis tal que $\forall\, y_i, y_j \in Y, y_i \equiv y_j \bmod \tilde{\pi}$ sse a sequência x_i associada a y_i pelo esquema de decodificação for igual a x_j que este associa a y_j.

Ocorrerá um "erro" se ao transmitirmos x_i recebermos y' tal que $x_i \neq x_j$, onde x_j é o símbolo associado a y' pelo esquema de decodificação. Ou seja, ao transmitirmos x_i a sequência y' recebida não pertence ao bloco da participação associado a x_i.

O esquema de decodificação que minimiza a probabilidade de erro a priori é o que $\forall y \in Y$ escolhe $x' \in X$ tal que $p(x'/y) \geq p(x/y), \forall x \in X$. Este esquema é chamado "observador ideal".

Como para $|x| \neq |y|$; $p(x/y) = 0$, bastará ao observador ideal escolher entre as sequências pertencentes a X de mesmo comprimento da sequência recebida.

Pela regra de Bayes temos que:

$$p(x/y) = \frac{p(x)\ p(y/x)}{p(y)},$$

onde p(y) é a probabilidade "a priori" de y, que só depende da fonte e do canal. Para maximizar p(x) p(y/x), basta maximizar p(x) p(y/x). Se tivermos o caso de sequências igualmente prováveis, $p(x) = \frac{1}{K}, \forall\, x \in X$.

Então:

$$p(x/y) = \frac{1}{K} \cdot \frac{p(y/x)}{p(y)} \qquad (3.2)$$

Vemos por 3.2 que no caso de sequências da fonte equiprováveis maximizar p(y/x) é equivalente a maximizar p(x/y). O esquema de decodificação que maximiza p(y/x) é o esquema "maximum likelihood" e neste caso é equivalente ao observador ideal.

3.3 Decodificação sintática

O que caracteriza a decodificação como sintática, além da fonte ser gramatical, é que o decodificador irá procurar dada a sequência y recebida, pela derivação mais provável. No caso extremo da sequência ser recebida sem erros o decodificador se reduzirá a um "parser".

Como consideraremos a fonte como uma gramática na forma padrão, a cada produção usada corresponderá um único terminal associado. Assim, o conjunto A não será T e sim J, o conjunto de rótulos de produções.

As probabilidades de transição serão pois

$$p(b_j/l_i) = p(b_j/a_i) \qquad (3.3)$$

O elemento a_i da equação 3.3 é o não terminal associado à produção l_i.

O conjunto de sequências da fonte será então

$X = L = \{ l \mid l \in J^* \text{ e } S \overset{l}{\Rightarrow}_p x \text{ e } x \in L_p(G) \}$, onde G é a gramática na forma padrão e $\overset{l}{\Rightarrow}_p$ é a derivação que obtemos aplicando as produções da sequência l.

O observador ideal, recebida a sequência y, a decodificará como l' tal que $p(l'/y) \geq p(l/y)$ $\forall$ $l \in L$.

Como as produções da gramática padrão estão na forma normal de Greibach, $p(l/y) = 0$ se $|l| \neq |y|$.

Analogamente, pela fórmula de Bayes:

$$p(l/y) = \frac{p(l)p(y/l)}{p(y)} \qquad (3.4)$$

Como p(y) independe de l, para maximizar p(l/y), basta maximizar p(l)p(y/l) (3.5)

Seja $l_1 = l_1 l_2 \ldots l_n$, $l^j = l^1 l^2 \ldots l^j$, onde $j \leq n$.

Por 2.1 teremos:

$$\begin{aligned} p(l) &\triangleq p_o(l_1)p(l_2/l_1)p(l_3/l_1 l_2) \ldots p(l_n/l_1 l_2 \ldots l_{n-1}) \\ &= p_0(l_1)p(l_2/l^1)p(l_3/l^2) \ldots p(l_n/l^{n-1}) \\ &= p_0(l_1) \prod_{i=2}^{n} p(l_i/l^{i-1}) \end{aligned} \qquad (3.6)$$

Seja a sequência recebida $y = b_1 b_2 \ldots b_n$ e $y^j = b_1 b_2 \ldots b_j$ com $j \leq n$.

Por 3.1 e 3.3 temos que:

$$p(y/l) = p(b_1/l_1)p(b_2/l_2) \ldots p(b_n/l_n) =$$

$$= \prod_{i=1}^{n} p(b_i/l_i) \qquad (3.7)$$

Substituindo 3.6 e 3.7 em 3.5, obtemos:

$$p(l)p(y/l) = p_0(l_1) \prod_{i=2}^{n} p(l_i/l^{i-1}) \prod_{i=1}^{n} p(b_i/l_i)$$

Fazendo $p(l_1/l^0) \triangleq p_0(l_1)$, teremos:

$$p(l)p(y/l) = \prod_{i=1}^{n} p(l_i/l^{i-1})p(b_i/l_i) \qquad (3.8)$$

3,4 Decodificador mínima distância:

Se as sequências da fonte de mesmo comprimento forem igualmente prováveis, o esquema de decodificação será, como vimos, o "maximum likelihood", ou seja, escolherá a derivação l que maximize p(y/l). Se além disso o canal for simétrico, a derivação l que maximiza p(y/l) é a que produz uma sequência de terminais em que a "distância de Hamming" de y seja mínima.

Dada duas sequências de mesmo comprimento u e v, a "distância de Hamming" d(u,v) entre essas sequências é o número de símbolos diferentes em cada posição da sequência.

Um decodificador "mínima distância" é o que associa à sequência recebida y a sequência x de menor distância de Hamming d(y,x), entre as possíveis sequências da fonte. Noccaso da decodificação sintática consideremos uma "derivação mínima distância", uma derivação que produz a sequência de terminais de menor distância da sequência recebida.

Exemplificando, consideremos o caso da gramática padrão que gera sequências refletidas num vocabulário terminal binário.

Exemplo 3.2 Seja a seguinte gramática na forma padrão:

G:

$N = \{S, S_0, S_1\}$

$T = \{1, 0, C\}$ com produções:

1) $S \rightarrow 0SS_0$ (1, 2, 3) (1/3, 1/3, 1/3)

2) $S \rightarrow 1SS_1$ (1, 2, 3) (1/3, 1/3, 1/3)

3) $S \rightarrow C$ S_0: (4) (1) S_1: (5) (1)

4) $S_0 \rightarrow 0$ S_0: (4) (1) S_1: (5) (1)

5) $S_1 \rightarrow 1$ S_0: (4) (1) S_1: (5) (1)

v_0 = (1/3 1/3 1/3 0 0), teremos então:

$L_p(G) = \{ uCu^T \mid u \in \{0,1\}^* \}$ e $x \in L_p(G)$

$p(x) = 1/3(2/3)^{\frac{x-1}{2}}$

o que nos mostra serem as sequências de mesmo comprimento da linguagem $L_p(G)$ equiprováveis. Se supormos o canal de transmissão simétrico, ou seja, com matriz de transição da forma:

$$\begin{bmatrix} p & \frac{1-p}{2} & \frac{1-p}{2} \\ \frac{1-p}{2} & p & \frac{1-p}{2} \\ \frac{1-p}{2} & \frac{1-p}{2} & p \end{bmatrix}$$

teremos que o observador ideal será o decodificador mínima distância.

CAPÍTULO IV

DECODIFICAÇÃO SINTÁTICA SEQUENCIAL

4.1 Introdução

A idéia de decodificação sintática sequencial (DSS) é aplicar as sequências da decodificação sequencial à decodificação sintática.

A decodificação sequencial é usada na teoria da informação para os códigos convolucionais. Assim como nos códigos convolucionais, é possível se construir uma árvore que nos dá todas as possíveis derivações da gramática padrão.

Construindo a árvore, o decodificador sequencial segue pelos nós com uma medida de se estar no nó certo. Assim que esta probabilidade cai abaixo de um certo valor ou limiar, o decodificador volta sôbre seus próprios passos e procurará uma derivação que não viole este limiar ("threshold"). O decodificador chega ao fim quando encontra um nó que corresponde a uma sequência transmitida de mesmo comprimento que a sequência recebida.

4.2 Árvore de derivações

Passemos então à construção formal de árvore sôbre a qual se buscará a decodificação sintática sequencial. Chamemos a esta nova árvore de "árvore de derivações".

Definição 4.1 Dada uma gramática $G = \langle N,T,P,S \rangle$, livre de contexto na forma normal de Greibach, uma "árvore de derivações do grau n" (ADG(n)) de G é uma árvore cujos nós satisfazem as condições:

1) Todo nó tem um "nome", que é uma sequência pertencente à V^*.

2) O nome da raiz é S (*).

3) Seja $x \in ADG(n)$, então $x = uv$, onde $u \in T^*$ e $v \in N^*$, se $x \Rightarrow y$ e $|u| < n$, então $y \in ADG(n)$.

4) Se $x, y, z \in ADG(n)$ e $x \Rightarrow y$ e $x \Rightarrow z$ então $y \neq z$.

5) Se $x, y \in ADG(n)$ e y é descendente imediato de x, então $x \Rightarrow y$.

Da definição da árvore de derivações temos que:

Teorema 4.1 Se $x, y \in ADG(n)$ e y é descendente de x, então $x \overset{*}{\Rightarrow} y$. Prova: Se y é descendente de x, então existe uma cadeia $u_0, u_1, \ldots, u_m$, onde $u_0 = x$ e $u_m = y$ tal que u_i é descendente imediato de u_{i-1}, $1 \leq i \leq m$, logo temos pela definição que $u_{i-1} \Rightarrow u_i \; \forall \, i$, $1 \leq i \leq m$, ou seja $x \overset{*}{\Rightarrow} y$.

Corolário 4.1 Se $y \in ADG(n)$ então $S \overset{*}{\Rightarrow} y$.
Prova: Imediata, pois se $y \in ADG(n)$ ou y é descendente de S ou é o próprio S. Em ambos os casos $S \overset{*}{\Rightarrow} y$.

Corolário 4.2 Se $y \in ADG(n)$ e $y \in T^*$ então $y \in L(G)$ e $|y| \leq n$.
Prova: Pelo corolário 4.1, temos que $S \overset{*}{\Rightarrow} y$, como $y \in T^*$,

(*) Para simplificar a notação, confundiremos de agora em diante o nó da árvore de derivações com seu nome.

então $y \in L(G)$. $|y| \leq n$ é fàcilmente demonstrável por absurdo. Suponhamos que $|y| > n$. Então $\exists \, x \in ADG(n)$, tal que $x \Rightarrow y$. Como a gra-

-mática está na forma normal de Greibach, temos que x=uv, onde u ∈ T* e v ∈ N*, e |u|= n. Vemos pelo item 4 da definição 4.1 que isto é impossível.

Este último corolário nos diz que se na árvore de derivações tivermos um nó composto só de terminais, então o nome deste nó pertence à linguagem gerada por G. Chamaremos a este nó de "nó terminal".

Exemplo 4.1 Seja a gramática G= ⟨ N,T,P,S ⟩ , onde N= { S,A } , T = {0,1} e P:

S → 0AA

S → 1S

S → 0

A → 1A

A → 0

A árvore de derivações do grau 3 desta gramática é dada pela figura 4.1.

A "distância" de um nó é o comprimento da lista de terminais do nó. A distância de um nó nos dá quantas produções foram usadas para atingir o nó.

Se considerarmos gramáticas no forma padrão, podemos construir "árvores de p-derivações" para estas gramáticas. Dada uma gramática G = ⟨ N,T,P,J,F ⟩ na forma padrão, a "árvore de p-derivações do grau n" desta gramática é a árvore de derivações do grau n de G' = ⟨ N,T,P,S ⟩ onde retiramos todas as derivações que não são também p-derivações. Para isto basta substituirmos na definição 4.1 derivação por p-derivações. Verifica-se facilmente que o teorema 4.1 e os corolários 4.1 e 4.2 se aplicam para a árvore de p-derivações.

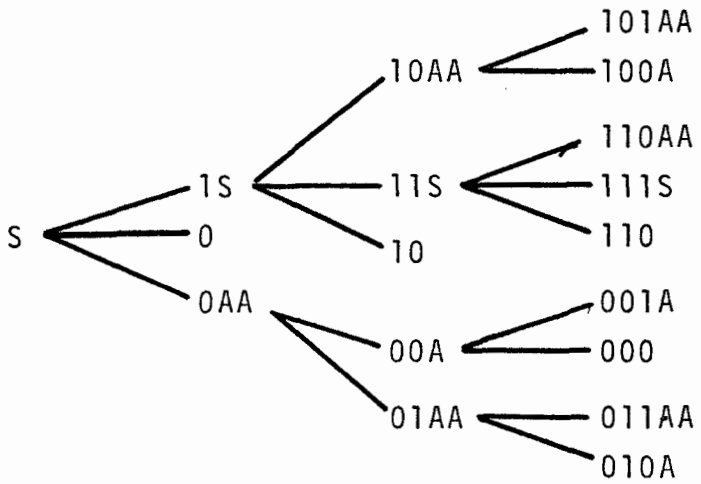


figura 4.1

Como só trataremos de gramáticas na forma padrão, ao falarmos de árvore de derivações para uma gramática, estaremos nos referindo à árvore de p-derivações.

Exemplo 4.2 Seja a gramática na forma padrão:

1) S → 0AA (4 5) (0.5 0.5)
2) S → 1S (2 3) (0.25 0.75)
3) S → 0 S: (1) (1) A: (4 5) (0.5 0.5)
4) A → 1A (4 5) (0.4 0.6)
5) A → 0 S: (1 3) (0.5 0.5) A: (5) (1)

Se o vetor inicializante v_0 for (0.6 0.4 0 0 0), teremos a seguinte árvore de derivações do grau 3.

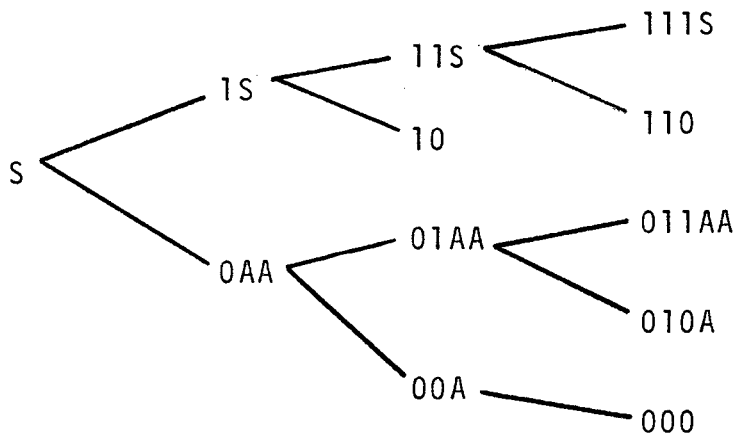


figura 4.2

4.3 A árvore dos valores dos nós

A idéia da decodificação sequencial é, como vimos, recebida uma sequência, decodificá-la símbolo por símbolo seguindo pelos nós da árvore acompanhado de uma medida da probabilidade de estarmos no nó certo.

Esta função que mede a probabilidade tem que satisfazer certas condições para que seja possível construir o decodificador baseado no princípio exposto acima. Primeiro devemos poder calcular esta função a partir dos nós já visitados, ou seja, a função não pode fazer referência a símbolos da sequência recebida que se encontrem além da distância do nó. A outra condição é que a função deve ter uma boa aproximação da probabilidade de estarmos no nó certo.

Recebida y, onde $|y| = n$, seja x um nó da árvore de derivações do grau n. Seja $l^i = l_1 l_2 \ldots l_i$ a sequência de rótulos de produções da derivação $S \overset{*}{\underset{p}{\Rightarrow}} x$.

Usaremos como medida aproximada de estarmos no nó certo, caso estejamos em x uma "fúnçao avaliadora" que chamaremos de "função-z" do nó e é dada pela equação 4.1.

$$z_x = - \log \frac{\prod_{j=1}^{i} w^j}{k^i}, \qquad (4.1)$$

onde $w^j = P(l_j/l^{j-1})P(b_j/l_j)$ (4.2)

$P(l_1/l^0)$ é dada pela equação (3.7a) e k^i é uma constante arbitrária (a ser determinada), elevada a i-ésima potência.

De 4.1 temos que:

$$z_x = - \sum_{j=1}^{i} \log \frac{w^j}{k}, \text{ ou seja} \qquad (4.3)$$

$$z_x = z^{i-1} - \log \frac{w^i}{k}, \qquad (4.4)$$

onde z^{i-1} é o valor da função-z do nó antecedente a x.

Dado o nó x da árvore de derivações, chamaremos a função-z deste nó de "valor" de x. A função-z é na realidade uma aproximação da probabilidade de estarmos no nó certo, pois $\prod_{j=1}^{i} w^j$ é uma função crescente de P(y/l). A finalidade da constante k será esclarecida noutra seção deste capítulo.

Para uma sequência recebida y, onde |y| = n, para todo nó da árvore de derivações do grau n, teremos um valor da função avaliadora. Assim, os valores de z induzem um mapeamento dos nós da árvore de derivações do grau n em uma "árvore dos valores dos nós".

Definição 4.3 A "árvore dos valores dos nós" é a árvore que obtemos ligando os pontos do gráfico da função avaliadora dos nós em função da distância do nó da seguinte forma: ligamos o ponto que representa o valor do nó aos pontos que representam os valores de seus descendentes e sucessores diretos. O valor da origem, z_S é por definição igual a zero.

Exemplo 4.3 Seja a gramática padrão:

1) S → 0AA (4 5) (0.5 0.5)

2) S → 1S (1 2 3) (0.2 0.3 0.5)

3) S → 0 A: (4 5) (0.4 0.6)

4) A → 1A (4 5) (0.4 0.6)

5) A → 0 A: (5) (1)

com vetor inicializante: (0.5 0.5 0 0 0) e o canal dado pela seguinte matriz de transição:

$$\begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} \begin{array}{cc} 0 & 1 \end{array} \\ \begin{bmatrix} 0.6 & 0.4 \\ 0.2 & 0.8 \end{bmatrix} \end{array}$$

Se recebermos a sequência 010, teremos a árvore dos valores dos nós dada pela figura 4.3

4.4 Algoritmo de decodificação sintática sequencial

O decodificador "olha" um nó da árvore dos valores de cada vez. Podemos imaginar que este nó é designado por um "ponteiro" que se move. O decodificador mantém ainda um "limiar corren-

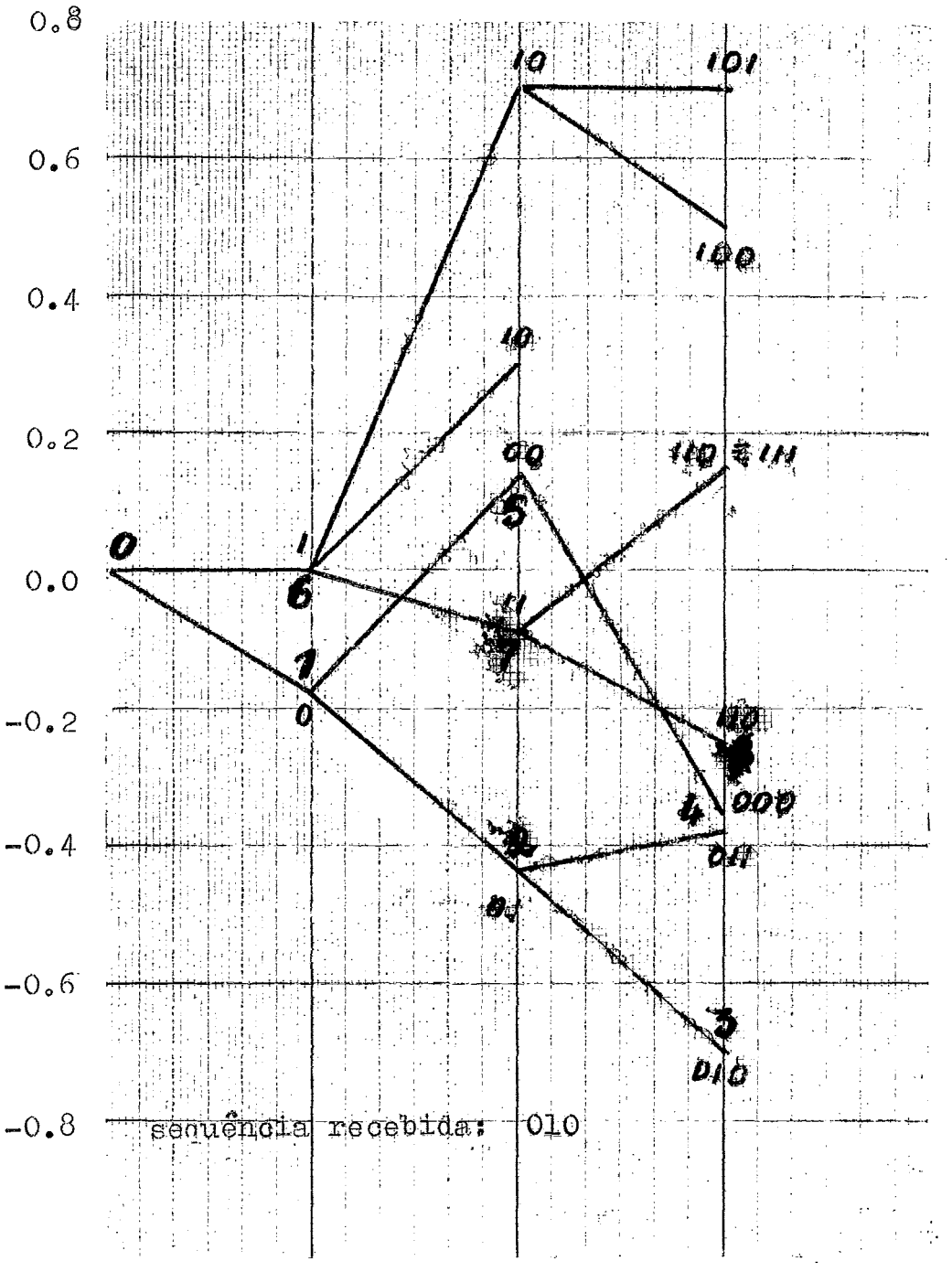


figura 4.3

te" T = nΔ, onde n é um número inteiro e Δ uma constante arbitrária.

Um nó da árvore dos valores é dito "satisfazer" todos os limiares acima e "violar" todos os que estão abaixo. Dos nós que divergem de um dado nó, o que tiver menor valor-z é chamado "melhor"e o que tem maior valor-z é dito "pior" nó.

Dizemos que o limiar é "abaixado" quando mudamos n de tal maneira que T seja o menor limiar satisfeito pelo nó apontado.

O algoritmo de decodificção que iremos descrever é um conjunto de regras para o movimento do ponteiro de um nó para outro da árvore dos valores. O algoritmo descrito é basicamente o algoritmo de Fano $(_{27},_{12})$, com as modificações necessárias para o caso da decodificação sintática.

As regras involvem o valor z_l do nó em que está o ponteiro e o valor z_{l+1} ou z_{l-1} do nó que se deseja alcançar e o limiar corrente T_l, onde l indica a distância do nó à raiz.

Regras para o movimento do ponteiro:

REGRAS	valor do nó olhado	valor do nó do ponteiro	movimento	limiar final	próximo nó a ser olhado	próxima regra a ser usada
1	$z_{l+1} \leqslant T_l$	$z_l > T_l - \Delta$	p/ frente	abaixe	melhor nó descendente	1 ∨ 2 ∨ 3
2	$z_{l+1} \leqslant T_l$	$z_l \leqslant T_l - \Delta$	p/ frente		melhor nó descendente	1 ∨ 2 ∨ 3
3	$z_{l+1} > T_l$				nó antecedente	4 ∨ 5
4	$z_{l-1} \leqslant T_l$		p/ trás		próximo melhor nó d/	1 ∨ 2 ∨ 3
5	$z_{l-1} > T_l$			suba Δ	melhor nó descendente	1 ∨ 2 ∨ 3

figura 4.4

Regras adicionais:

i. o ponteiro é colocado inicialmente na origem, com $z_0=T_0=0$, olhando para o melhor nó descendente.

ii. é suposto existir um nó imaginário anterior à origem, com valor $z_{-1}=+\infty$. Isto faz com que o decodificador ao chegar à origem por um movimento para trás vá necessariamente para frente, aumentando o limiar de (regra 5.)

iii. analogamente, faremos corresponder a cada nó um descendente imaginário de valor $+\infty$. Pela regra 3 vemos que ao atingir este nó, o ponteiro olhará para o nó antecedente ao que estiver.

Observações: 1) Temos pelas regras de movimento que só é permitido ao ponteiro se movimentar se o próximo nó visitado não violar o limiar corrente.

2) A regra adicioanl 3 nos impede de tentar seguir pela árvore ao encontrarmos um nó terminal de distância menor que n (comprimento da sequencia recebida), ou ao chegarmos à distância n, o nó apontado não ser terminal.

Antes de fazermos as idéias mais precisas, são necessários alguns conceitos e definições adicionais.

Um nó é "visitado" pelo ponteiro quando é apontado por este e é "F-visitado" quando é alcançado através da aplicação das regras 1 e 2, ou seja, resultante de um movimento para frente.

Estando o ponteiro do nó x, com distância igual a 1, os "antecedentes" de x são os nós que ligam x com a origem. Os "valores do caminho" T_0, T_1, ..., T_1 associados a x são os li-

miares finais das visitas mais recentes aos antecedentes de x. Os "descendentes" de x são os nós para os quais x é um antecedente. Os "descendentes imediatos" de x são os nós descendentes de x ligados diretamente a x.

O teorema 4.2 nos descreve o movimento do ponteiro, ou seja, a operação do algoritmo. A demonstração deste teorema pode ser encontrado no apêndice A.

Teorema 4.2 a) Se o ponteiro estiver no nó x, com distância 1, os limiares do caminho, x $T_0, T_1, T_2, \ldots, T_l$ e os valores do caminho $z_0, z_1, z_2, \ldots, z_l$ satisfazem as seguintes desigualdades:

a.1 $T_i \geq z_i \qquad 0 \leq i \leq l$

a.2 $T_{i+1} \leq T_i \qquad 0 \leq i \leq l-1$

a.3 $T_{i-1} \leq T_i - \Delta \Rightarrow T_i - \Delta < z_i$

a.4 $T_{i+1} \leq T_i - \Delta \Rightarrow T_{i+1} < z_i$

b) Para todo nó F-visitado, o limiar final T da primeira F-visita é n acima de T, e nas F-visitas subsequentes o limiar final é acima do limiar finalda visita anterior.
obs: $n = \frac{T'-T}{\Delta} - 1$, onde T' é o limiar mais baixo tal que $T' \geq z'$, sendo z' o valor do nó antecedente se $T' < T$. Caso $T' \geq T$, então $n = 1$.

c) Seja x F-visitado com limiar final T. Então, antes que x seja visitado outra vez, todo descendente de x para o qual o caminho de x está abaixo de T será F-visitado com limiar

final T. Além disto, entre a visita a x e a próxima, o limiar não será aumentado para um limiar acima de T.

A figura 4.5 nos mostra a situação dos valores do caminho e os limiares associados a um nó x.

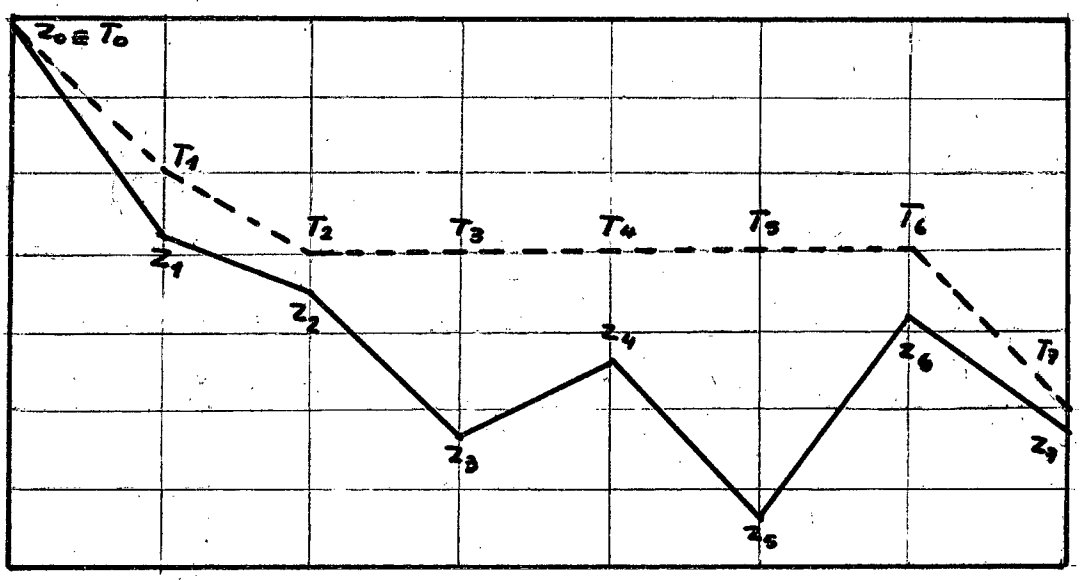


figura 4.5

O funcionamento do algoritmo é melhor entendido através um exemplo. Consideremos a árvore de valores do exemplo 4.1, figura 4.3, com $\Delta = 0.4$. O seguinte quadro ilustra os passos do algoritmo até a decodificação final.

PONTEIRO	LIMIAR	NÓ OLHADO	VALOR DO NÓ BASE	REGRA A SER USADA	VÁ PARA :	PRÓXIMO LIMIAR	PRÓXIMA REGRA
0	0	1	>	1	1	0	A
1	0	2	>	1	2	-Δ	A
2	-Δ	3	>	1	3	-Δ	A
3	-Δ	I X ***		3			B
3	-Δ	2		4	2	-Δ	A
2	-Δ	4 X		3			B
2	0	1 X		5		0	A
2	0	3	≤	2	3		A
3	0	I X		3			B
3	0	2		4	2		A
2	0	4	≤	2	4		A
4	0	I X		3			B
4	0	2		4	2		A
2	0	I X		3			B
2	0	1		4	1		A
1	0	5 X		3			B
1	0	0		4	0		A
0	0	6	>	1	6	0	A
6	0	7	>	1	7	0	A
7	0	8	>	1	8	0	FIM

figura 4.6

Observações: (*) Nesta coluna compara-se o valor do nó base, isto é, o nó apontado pelo ponteiro com T- , onde T é o limiar vigente.
(**) A = 1,2,3 e B = 4,5 .
(***) I indica um nó imaginário com valor igual a e X indica violação do limiar corrente.

Como consequências do teorema 4.2, temos os seguintes teoremas que nos garantem o "bom" funcionamento do algoritmo de decodificação. O teorema 4.3 nos diz que o decodificador não entra em "loop", e o teorema 4.4 que se houver pelo menos um nó terminal de distância igual ao grau da árvore de derivações, então o decodificador chegará ao fim.

Teorema 4.3 Dada uma árvore de valores dos nós, o ponteiro do algoritmo de decodificação não visita duas vezes o mesmo nó com o mesmo limiar e olhando para o mesmo nó. Isto quer dizer que o mesmo estado não se repete durante a decodificação de uma sequência, o que implicaria em um "loop", já que o algoritmo é determinístico.

Prova: Como pelo teorema 4.2, parte b, duas F-visitas a um nó não tem o mesmo limiar final, logo duas visitas só poderiam ter o mesmo limiar final caso acontecessem sem um F-visita entre elas. Então temos que uma das duas visitas deve ser oriunda da regra 4. Por esta mesma regra, o nó olhado é o próximo melhor nó, eliminando-se assim a possibilidade de se olhar um nó já olhado anteriormente.

Teorema 4.4 Se na árvore dos valores houver pelo menos um nó terminal com distância igual ao grau da árvore, o algoritmo chegará ao fim.

Prova: Chamemos de "nó final" ao nó terminal de distância igual ao grau da árvore de derivações, ou seja, igual ao comprimento da sequência recebida.

- seja f um nó final. Então os valores do caminho associados a f são todos finitos, pois por hipótese o valor de f é finito.
- seja x_1 o nó antecedente a f cuja distância é igual a 1 e x_m o nó antecedente de maior valor entre todos os nós do caminho.
- seja x_1 F-visitado na primeira vez com limiar final T_1^1. Se $z_m \leq T_1^1$, o nó x_m será F-visitado com limiar final T_1^1 antes que x_1 seja novamente visitado, e por 4.2, c, temos que f será F-visitado e portanto decodificado.
- como toda vez que x_1 é F-visitado o limiar final T_1^j da j-ésima F-visita satisfaz a desigualdade: $T_1^j > T_1^{j-1}$ por 4.2-b, para algum j temos $T_1^j \geq z_m$. Quando isto acontecer, f será visitado.

Resta provar que x_1 será visitado contínuas vezes a não ser que o decodificador chegue a um nó final diferente de f.

- seja $z_{11}, z_{12}, \ldots z_{1n}$ os valores dos nós de distância 1 tal que $z_{1i} \leq z_{1j}$ para $j > i$, e $z_{1i} \leq z_1$ $1 \leq i \leq n$. Se nenhum descendente dos nós z_{1i} for um nó final, então haverá

sempre nós descendentes de z_{1i} cujos valores z são tais que $z > T, \forall T$. Por 4.2-c, se z_{1i} for F-visitado com limiar final T_i depois que todos os nós de valor menor que T_i forem visitados, o ponteiro volta a z_{1i} e tentará F-visitar z_{1i+1}. Ou seja, os nós z_{1i} serão todos visitados na ordem crescente de seus valores. Então x_1 será F-visitado pela primeira vez. Pelas mesmas razões, x_1 será F-visitado seguidamente até que f seja alcançado, chegando o decodificador ao fim.

4.5 Exame da função-z dos nós

Um nó x do caminho correto, de valor z, é dito ser um nó de "quebra", se $z \leq T_i$, onde T_i são os limiares do caminho dos descendentes de x associados ao nó final correto.

Pelo teorema 4.2, parte b, o limiar final T da primeira F-visita a x satisfaz $T-\Delta < z \leq T$. De 4.2-c, temos que se x é um nó de quebra, não será visitado de novo até que todos os nós do caminho correto seja visitado, ou seja, x não será nunca visitado novamente.

Se supormos que todos os caminhos incorretos "flutuam" para cima, o decodificador possivelmente decodificará o nó final correto.

O problema então é fazer com que os valores do caminho que levam ao nó final correto flutuem para baixo e os que levam a nós finais incorretos flutuem para cima.

Isto é conseguido através da própria função z dos nós e da constante k que permaneceu até aqui inexplicada.

Se considerarmos os valores dos nós finais teremos:

$$z_f = -\sum_{j=1}^{n} \log \frac{w^j}{k}$$

$$z_f = -\log \frac{\prod_{j=1}^{n} P(l_j/l^{j-1})P(b_j/l_j)}{k^n} \qquad (4.5)$$

Mas por 3.8 temos:

$$j = \prod_{1}^{n} P(l_j/l^{j-1})P(b_j/l_j) = P(l)P(y/l), \qquad (4.6)$$

onde y é a sequência recebida.

Como vimos anteriormente, o lado direito da equação 4.6 é a expressão que o observador ideal procura maximizar variando l.

Como k é uma constante, o nó final de menor valor corresponde à derivação que seria decodificada pelo observador ideal, e que faremos corresponder ao que chamaremos de "nó correto".

A constante k deve ser escolhida para cada gramática padrão de tal forma que k^n esteja entre o maior e o menor valor de nó final. Assim, para os nós finais diferentes do correto o valor $\frac{P(l)\,P(y/l)}{k^n} < 1$ e o nó correto $\frac{P(l)\,P(y/l)}{k^n}$ seja o maior possível.

Ao fazermos isto, o que é sempre possível, o valor do nó correto será negativo e o menor possível enquanto os valores dos nós incorretos serão positivos. Ou seja, os valores do caminho do nó correto flutuarão para baixo e os dos nós incorretos flutuarão para cima. Evidentemente não é possível escolher um k

diferente para cada sequência recebida. O que fazemos é escolher um só k que satisfaça as condições acima para o maior número possível de sequências, ajustando o valor de k empiricamente, de tal modo que minimize a porcentagem de sequências decodificadas erradamente.

A observação que resta fazer é que o nó decodificado pelo observador ideal não corresponde necessariamente à sequência que foi transmitida, mas é a melhor estimativa que se pode fazer desta.

Para vizualizar melhor o procedimento do decodificador, podemos imaginar uma gramática padrão cujas probabilidades de transição das produções seja constante, como no exemplo 3.2, e um canal de transmissão simétrico.

Seja o nó x de distância i, descendente direto de y de valor z_{i-1}. Então teremos:

$$z_i = z_{i-1} - \log \frac{P(l_i/l^{i-1})\ P(b_i/l_i)}{k}$$

Como supomos $P(l_i/l^{i-1})$ constante, podemos fazer

$k = P(l_i/l^{i-1})k$, então teremos:

$$z_i = z_{i-1} - \frac{\log P(b_i/l_i)}{k'}$$

Como o canal é simétrico, temos:

$$P(b_i/l_i) = \begin{cases} p \text{ se } b_i = a_i, \text{ onde } a_i \text{ é o terminal da produção } l_i \\ q \text{ se } b_i \neq a_i \end{cases}$$

Seja $p > q$. Fazendo $k' = \bar{p}$, temos:

$$\log\left(\frac{p}{k'}\right) = 0 \quad \text{e} \quad \log\left(\frac{q}{k'}\right) = -\delta$$

$z_i = z_{i-1} + f(b_i, l_i)$, onde:

$$f(b_i, l_i) = \begin{cases} 0 & \text{se } b_i = a_i \\ +\delta & \text{se } b_i \neq a_i \end{cases}$$

Como $z_0 = 0$, podemos escrever que:

$z_i = d(y^i, x_i)\delta$, onde $d(y^i, x_i)$ é a distância de Hamming entre a subsequênica y^i da sequência recebida y e x_i que é a sequência de terminais do nó x.

Se fizermos $\Delta = \delta$, o algoritmo será simplificado da seguinte maneira:

1) $z_{i-1} \leq z_i$, o que fará que quando o ponteiro olhar para trás (regras 4 e 5), o limiar não será violado e o ponteiro voltará ao nó anterior, ou seja, a regra 5 só será necessária na origem.
2) Não haverá necessidade também da regra 1 pois o limiar não será nunca abaixado.

O limiar só é aumentado quando o ponteiro atinge a origem num movimento para trás. Isto significa que o algoritmo pesquizará todos os nós da árvore que satisfazem o limiar corrente antes de aumentá-lo. Como a função-z no caso é a própria distância de Hamming, a sequência decodificada será a de "mínima distância" da recebida. Neste caso o decodificador mínima distância é equivalente ao observador ideal.

Uma tal estratégia de decodificação (mínima distância) quando nem a fonte nem o canal satisfazem as exigências para haver equivalência com o observador ideal, representa ignorar as

características da fonte e do canal.

Na simulação do decodificador sintático sequencial, foi considerada também esta estratégia para se poder avaliar quanto se ganha em tempo e eficiência ao se levar em conta as características da fonte e do canal. Os resultados desse estudo comparativo são mostrados e analizados no capítulo V.

CAPITULO V

RESULTADOS

5.1 Introdução:

Neste capítulo são apresentados os resultados da simulação do algoritmo de decodificação sintática sequencial. É feita a comparação entre o "decodificador mínima distância" e o "decodificador sintático sequencial", que respectivamente abreviaremos por "DMD" e "DSS".

Ao usarmos o algoritmo que decodifica as sequências recebidas segundo sequências de mínima distância das sequências recebidas, ignoramos as características da fonte e do canal de transmissão, o que não acontece com o DSS. A comparação nos mostrará pois quanto ganhamos ao considerar estas características.

A comparação entre os dois decodificadores será feita em dois aspectos principais. Dados uma fonte (gramática padrão) e um canal de transmissão, comparamos a "eficiência" do decodificador, ou seja, a percentagem de sequências decodificadas corretamente, e o "tempo" decorrido para a decodificação de uma sequência recebida. Como medida deste tempo gasto, tomaremos o número de passos para trás (backtrackings) feitos na decodificação de uma sequência. Procurou-se, ao se tomar o número de "backtrackings" uma medida que independesse do computador e programa utilizados. O número de passos feitos durante a decodificação é achado multiplicando-se por 2 o número de "backtrackings" e somando o coprimento da sequência.

Na simulação foram usados três tipos de gramática. A primeira é a que gera a linguagem $L = \{ wCw^T \text{ onde } w \in \{0,1\}^* \}$. A se-

gunda gera a linguagem formada por senteças do cálculo proposicional. Finalmente a terceira gramática gera sentenças do portugues sobre um vocabulário limitado.

5.2 Resultados

A primeira gramática padrão utilizada como fonte no sistema de transmissão é dada pelas deguintes produções:

1) $S \rightarrow 0SS_0$ (1 2 3) (0.3 0.6 0.1)
2) $S \rightarrow 1SS_1$ (1 2 3) (0.6 0.3 0.1)
3) $S \rightarrow C$ S_0:(4) (1) S_1:(5) (1)
4) $S_0 \rightarrow 0$ S_0:(4) (1) S_1:(5) (1)
5) $S_1 \rightarrow 1$ S_0:(4) (1) S_1:(5) (1)

com vetor inicializante $v_0=(0.5\ 0.5\ 0\ 0\ 0)$.

A linguagem gerada por esta gramática é:

$L=\{wCw^T$ onde $w\epsilon\{0,1\}^*\}$, com probabilidades associadas às sequências pertencentes à linguagem.

O canal de transmissão é dado pela seguinte matriz de teansição:

	0	1	C
0	0.8	0.15	0.05
1	0.05	0.8	0.15
C	0	0	1.0

Observações:

1) O fato de ser determinística a transmissão do símbolo 'C' justifica-se porque as sequências nas quais o símbolo 'C' é transmitido com erro, o tempo de decodificação é demasiadamente grande para o DMD.

2) No DSS foi estudada a variação do "desempenho" (eficiência e tempo) com a variação do parâmetro .

3) Como medida da dispersão do número de "backtrackings" em torno da média, foi calculado o desvio padrão para cada comprimento de sequência.

— Na figura 5.1 é mostrada a variação da eficiência com o comprimento da sequência e nas figuras 5.2 e 5.3 o número médio de "backtrackings" e o desvio padrão respectivamente para cada comprimento de sequência.

A segunda gramática padrão utilizada gera sentenças do cálculo de proposições, e é dada pelas produções:

1) S → ¬ S (1 2 3 4 5) (0.1 0.2 0.3 0.1 0.3)
2) S → (SUS$ (1 2 3 4 5) (0.3 0.1 0.2 0.3 0.1)
3) S → (SES$ (1 2 3 4 5) (0.2 0.3 0.1 0.1 0.3)
4) S → x U:(6) (1) E:(7) (1) $:(8) (1)
5) S → y U:(6) (1) E:(7) (1) $:(8) (1)
6) U → v S:(1 2 3 4 5) (0.1 0.2 0.4 0.1 0.2)
7) E → ε S:(1 2 3 4 5) (0.4 0.1 0.2 0.2 0.1)
8) $ →) U:(6) (1) E:(7) (1) $:(8) (1)

com vetor inicializante v_0= (o.18 o.4 0.4 0.01 0.01 0 0 0)

O canal de transmissão é dado pela matriz de transição:

	x	y	v	ε	¬	)	(
x	0.9	0.02	0.03	0.02	0.01	0.01	0.01
y	0.03	0.90	0.01	0.02	0.02	0.01	0.01
v	0.01	0.01	0.90	0.02	0.03	0.02	0.01
ε	0.02	0.01	0.02	0.90	0.03	0.01	0.01
¬	0.02	0.01	0.03	0.02	0.90	0.01	0.01
)	0.01	0.01	0.02	0.02	0.01	0.90	0.03
(	0.01	0.01	0.02	0.02	0.01	0.03	0.90

Observação: As observações 2 e 3 referentes à primeira gramática também se aplicam a esta.

Na figura 5.4 temos a variação da eficiência e na figura 5.5 a variação do número de "backtrackings" com o comprimento das sequências e na 5.6 o desvio padrão.

A terceira gramática que usamos gera sentenças do portugues. As produções são as seguintes:

1) S → outroHAP (11 12) (0.5 0.5)
2) S → uma FAP (13 14) (0.5 0.5)

falta pág. 47 → erro de datilografia.

3) S → osHZQ (11 12) (0.5 0.5)
4) S → estasFZQLS (13 14) (0.5 0.5)
5) P → mordiaOAM (9 10) (0.5 0.5)
6) P → riaAM (15 16 17) (0.3 0.3 0.4)
7) Q → viamO (9 10) (0.5 0.5)
8) Q → comiamOAM (9 10) (0.5 0.5)
9) O → todasFZ (13 14) (0.5 0.5)
10) O → algumaHA (11 12) (0.5 0.5)
11) H → eunuco A:(15 16 17) (0.3 0.3 0.4) Z:(20) (1)
12) H → padre A:(15 16 17) (0.3 0.3 0.4) Z:(20) (1)
13) F → banana A:(15 16 17) (0.3 0.3 0.4) Z:(20) (1)
14) F → macaca A:(15 16 17) (0.3 0.3 0.4) Z:(20) (1)
15) A → feroz P:(5 6) (0.5 0.5) L:(18) (1) M:(19) (1)
16) A → cruel P:(5 6) (0.5 0.5) L:(18) (1) M:(19) (1)
17) A → soez P:(5 6) (0.5 0.5) L:(18) (1) M:(19) (1)
18) L → ja que S:(1 2 3 4) (0.3 0.3 0.2 0.2)
19) M → mente L:(18) (1)
20) Z → s Q:(7 8) (0.5 0.5) L:(18) (1) A:(15 16 17)
(0.3 0.3 0.4)

O canal de transmissão foi feito simétrico com 24 símbolos de entrada e de saída (23 letras do alfabeto e o 'branco') com probabilidade de trasição correta igual a 0.76 e incorreta de 0.01.

A decodificação de sequências desta terceira gramática é diferente das duas primeiras no fato que os elementos das sentenças são por sua vez sequências de símbolos.

Para o cálculo da função avaliadora dos nós (equação 4.2) foi tomado ao invés de $p(b_j/l_j)$ o produtório $\prod_{i=1}^{n} p(b_i/l_i^j)$, onde b_i, l_i, i=1, ... , n são respectivamente as letras da palavra recebida e as letras da palavra associada à produção l^j e n é o comprimento da palavra.

O fato de cada elemento da sentença ser uma sequência de símbolos, nos dá uma flexibilidade ao DMD que até agora não tínhamos. Podemos ao invés de aumentar o limiar corrente de 1 em 1 fazê.lo de um número inteiro maior que 1, sem que isto altere a eficiência do algoritmo. Isto se justifica ao sabermos que a distância mínima entre palavras do vocabulário é maior que1. Deste modo estudamos o desempenho do DMD para valores de que vão de 1 a 6.

Para esta gramática , preferimos dar a percentagem total de sequências decodificadas corretamente em lugar da variação da eficiência com o comprimento da sequência.

Na figura 5.7 estão as percentagens de sequências decodificadas corretamente para o DSS e DMD. Nas figuras 5.8 e 5.9 estão o número médio de "backtrackings" para o DMD e o desvio padrão respectivamente para os diversos Δ s.

Para o DSS não foi levantado o gráfico do número médio de "backtrackings" porque este foi praticamente igual a zero para todos os Δs.

5.3 Análise dos resultados:

As eficiências de decodificação tanto para a primeira quanto para segunda gramática são praticamente equivalentes para os dois decodificadores e não varia significativamente com Δ para o DSS:

Notamos também para as duas gramáticas que a eficiência de decodificação tende para zero quando o comprimento das sequências cresce. Acreditamos que isto se deva ao fato de que as sequências pertencentes às linguagens geradas pelas duas gramáticas se apresentam muito "próximas" umas das outras.

Para o caso da primeira gramática, ou seja a que gera $L=\{wCw^T$ onde $w \in \{0,1\}^*\}$, par a toda sequência x L, existe $n=|w|$ de sequências $y_1, y_2, \ldots y_n \in L$, onde $|y_i| = |x|$ e de distância de Hamming $d(y_i,x)=2$.

Para o caso da segunda gramática (cálculo proposicional)

temos também que a efiência tende para zero quando o comprimento das sequências aumenta. A explicação é a mesma que a dada para a primeira gramática.

Em toda sequência x pertencente à linguagem gerada pela segunda gramática, para cada símbolo diferente de '(' e ')', temos uma sequência y pertencente à linguagem de distância de Hamming d(y,x)=1.

Quanto ao número de "backtrackings", notamos uma grande diferença no número medio de "backtrackings" para as duas primeiras gramáticas, sendo esta diferença maior para a primeira gramática. Astaxas de crescimento do número médio de "backtackings" são também diferentes nas duas gramáticas, sendo um pouco maior para o DMD.

Os resultados obtidos para a terceira gramática devem ser analizados separadamente tendo em vista a natureza diferente dos dados obtidos.

O primeiro fato a ser notado é a pequena percentagem de sequências decodificadas erradamente, tanto para o DSS quanto para o DMD: Isto se deve a que udamos um vocabulário muito restrito, tornando difícil para o canal de transmissão "transportar" a sequência enciada para uma mais próxima de outra sequência também pertencente à linguagem. Isto reforça as explicações dadas para as duas primeiras gramáticas. Por esta mesma razão foi que pudemos usar para o DMD incrementos do limiar diferentes de 1 sem alterar a eficiência de decodificação.

Para o DSS a variação de Δ não alterou significativamente nem o número médio de "backtrackings" nem a eficiência de decodificação.

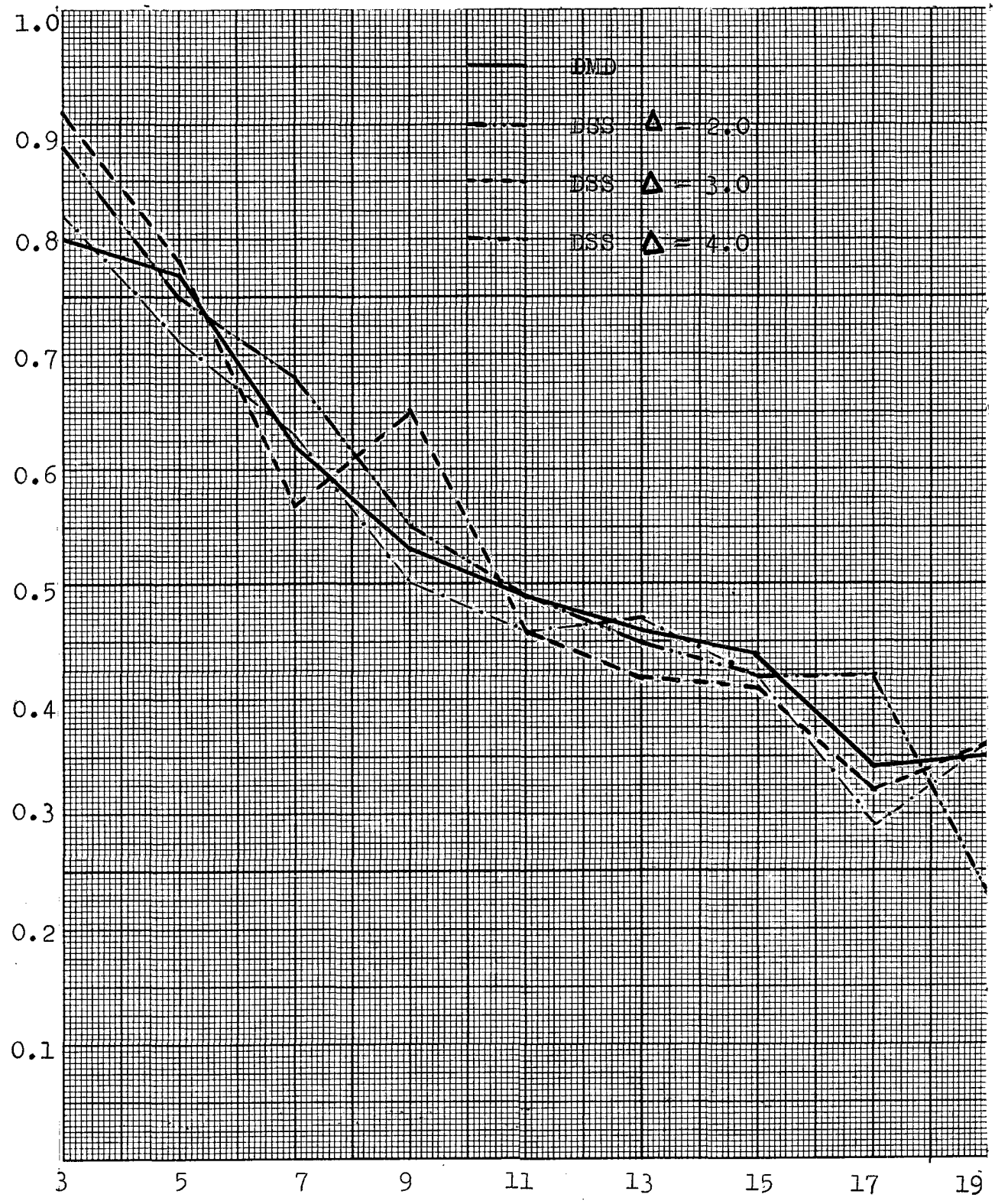

1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
3
5
7
9
11
13
15
17
19
DMD
DSS Δ = 2.0
DSS Δ = 3.0
DSS Δ = 4.0

figura 5.1

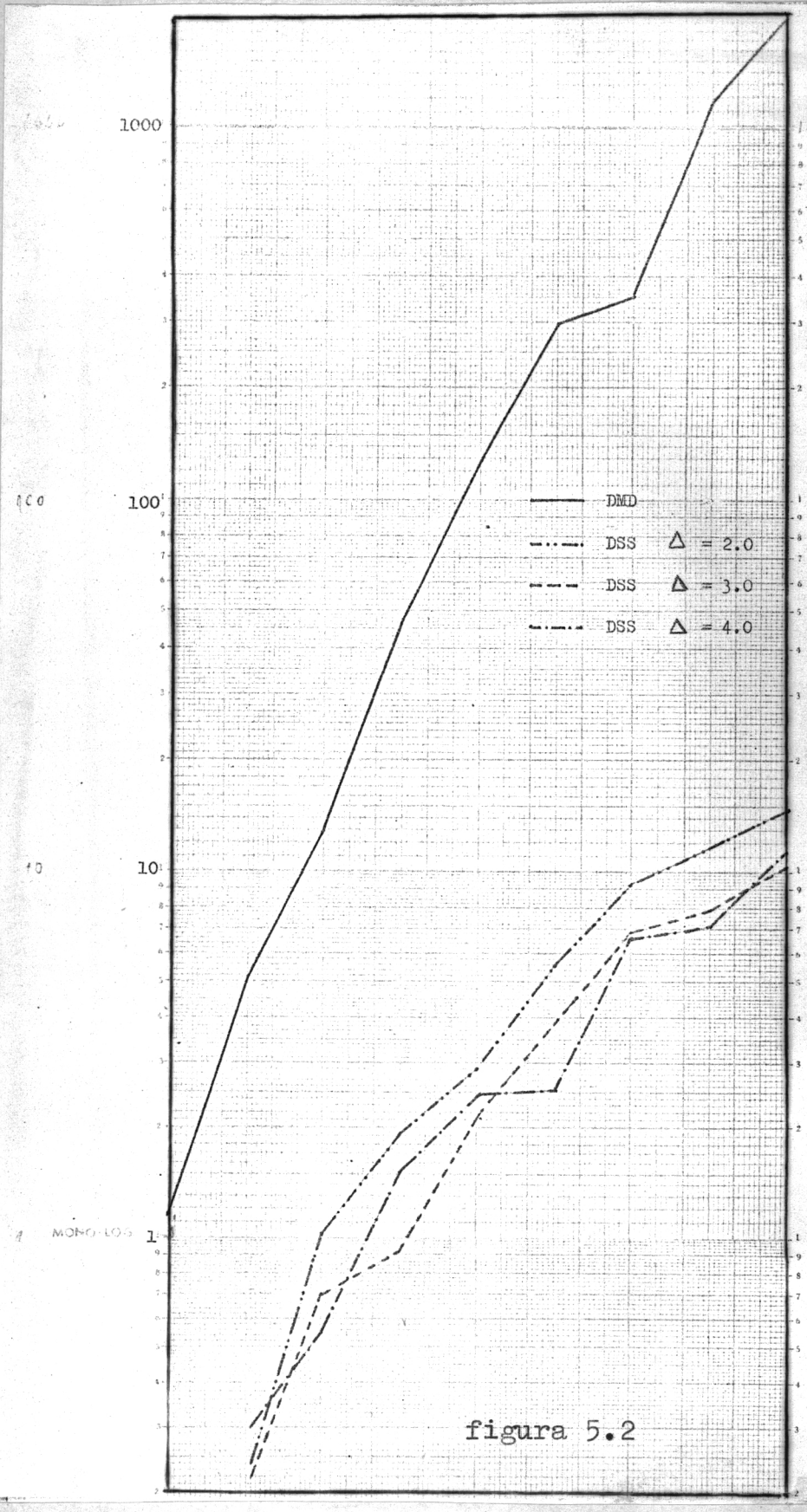


figura 5.2

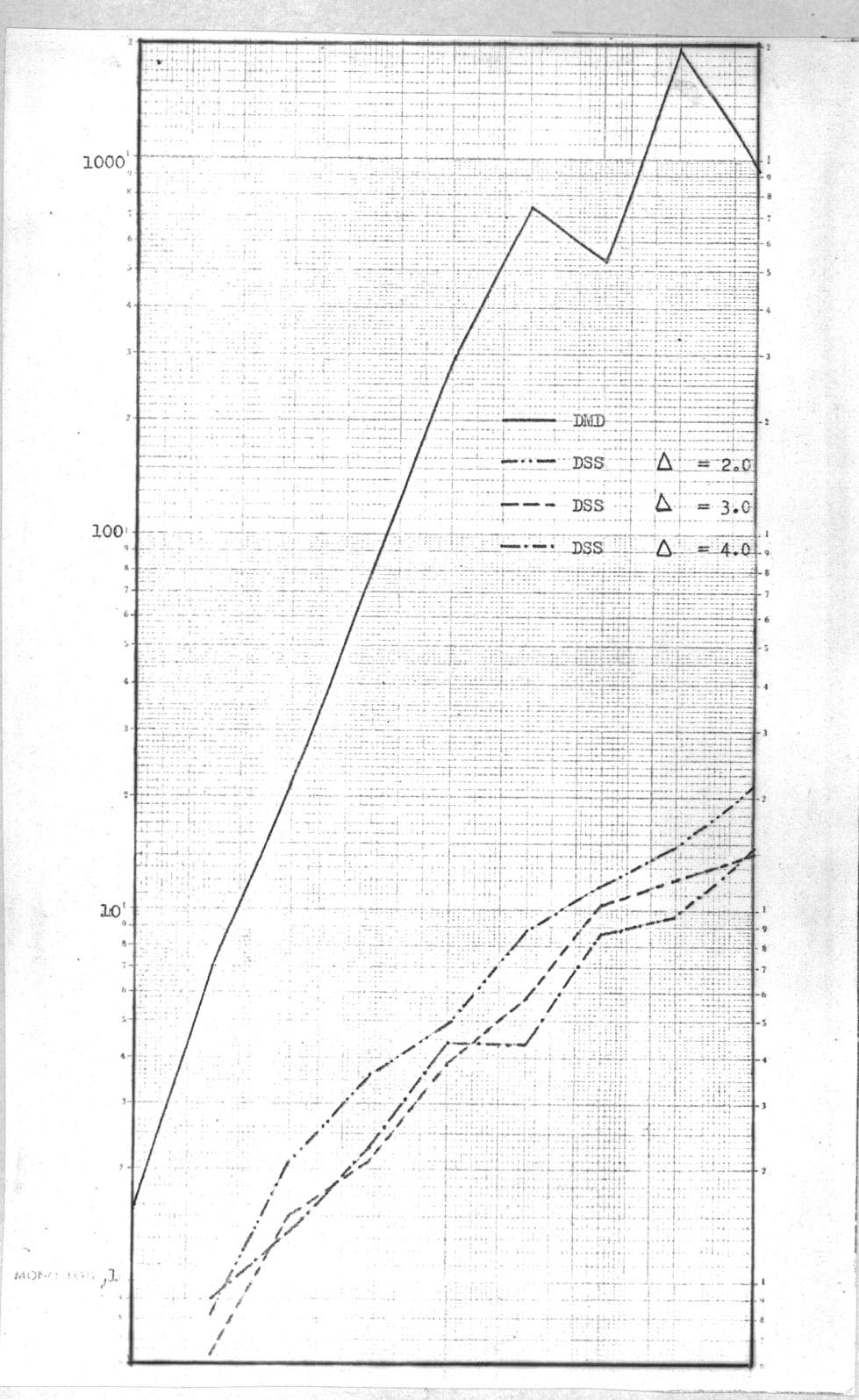


figura 5.3

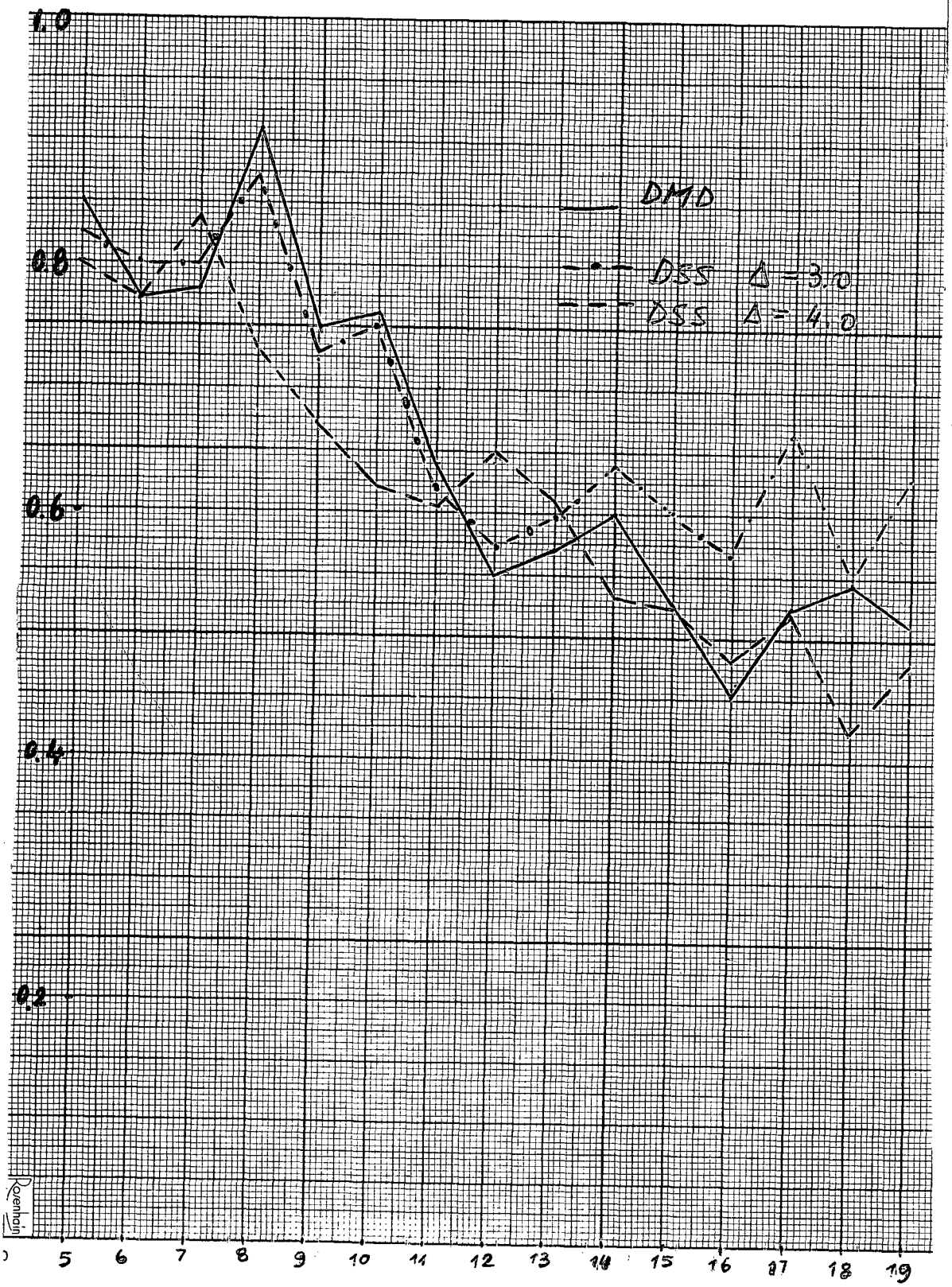

1.0
0.8
0.6
0.4
0.2
DMD
DSS Δ = 3.0
DSS Δ = 4.0
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19

figura 5.4

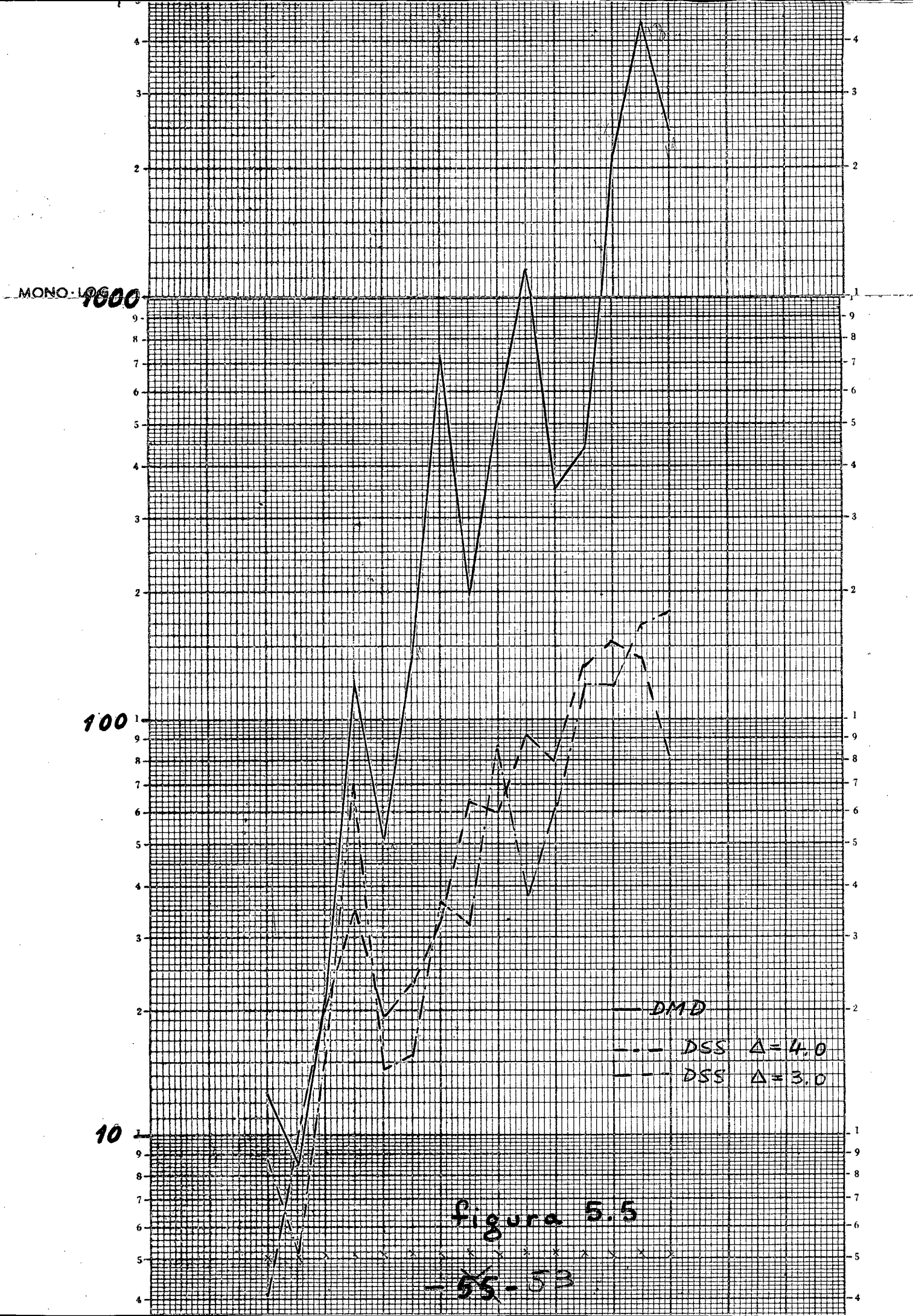


figura 5.5

Este gráfico não pode ser levantado em tempo por ter esse quebrado o computador.

figura 5.6

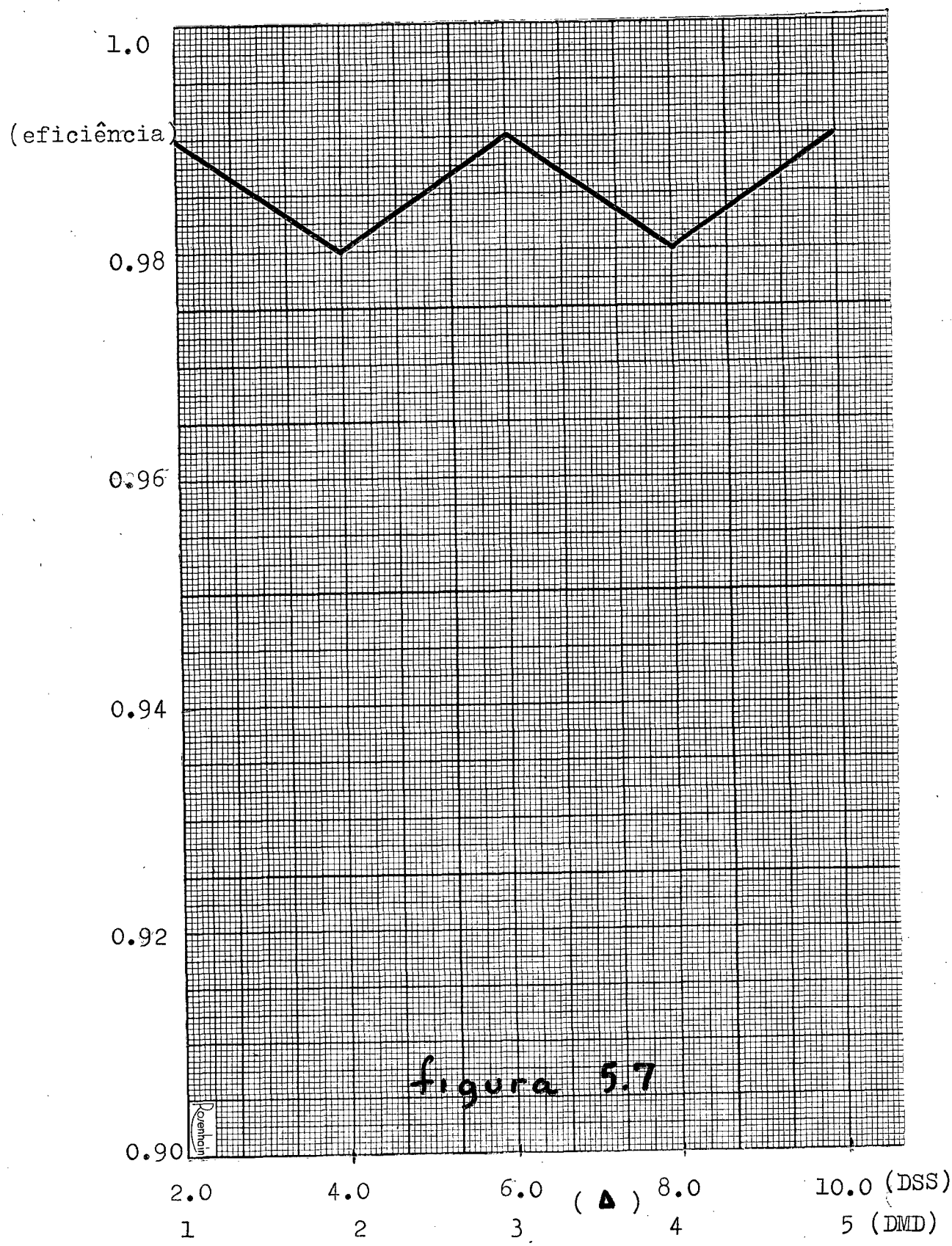

1.0
(eficiência)
0.98
0.96
0.94
0.92
0.90
2.0
1
4.0
2
6.0
3
(Δ)
8.0
4
10.0 (DSS)
5 (DMD)

figura 5.7

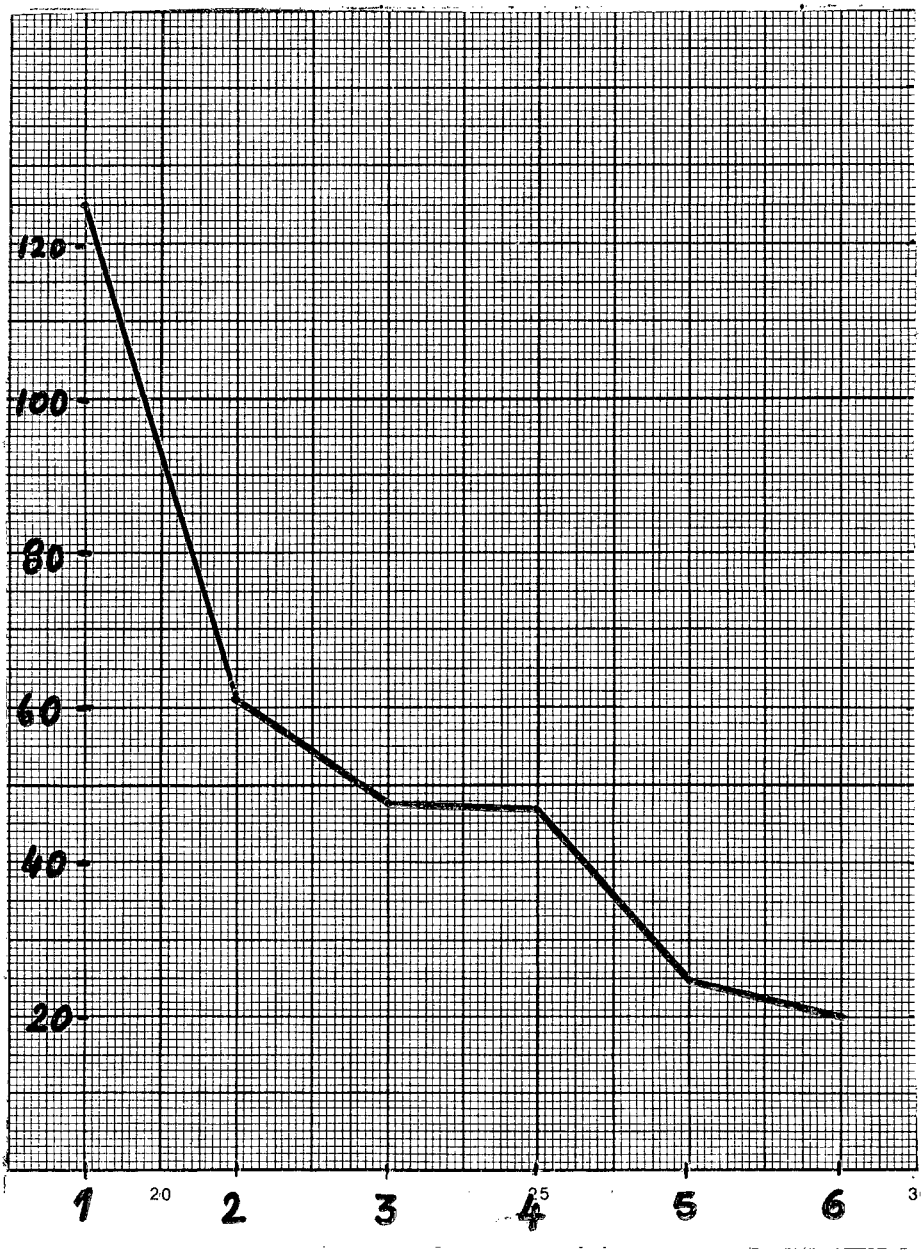


figura 5.8

CAPITULO VI

CONCLUSÕES

É estimado para um futuro próximo que uma grande parte, se não a maior parte do tráfego de comunicações consistirá da transmissão de informação entre computadores ou entre usuários e computadores (*18*). Já que grande parte desta comunicação é feita numa linguagem "formal", parecemnos natural estudar sistemas de comunicação adaptados para este tráfico. Deste modo podemos aproveitar a informação "sintática" contida em mensagens escritas nessas linguagens para ajudar na decodificação dessas mensagens, aumentando assim a confiabilidade da transmissão.

Em nosso estudo procuramos mostrar a aplicabilidade da idéia da decodificação sequencial a sistemas de transmissão nos quais a fonte é gramatical.

O esquema sequencial de decodificação é basicamente um "top-down parser" com a vantagem de que a cada ponto o decodificador só processa um único símbolo da sequência recebida. Isto faz com que os decodificadores sequenciais se adaptem com naturalidade a sistemas de transmissão, já que a sequência pode ser decodificada ao mesmo tempo que é recebida.

Na simulação do algoritmo de decodificação sequencial foi estudada a influência do comprimento das sequências da fonte e do parametro Δ na eficiência (porcentagem de sequências decodificadas corretamente) e no tempo de decodificação (número de "backtrackings") para o decodificador sintático sequencial (DSS) e o decodificador sequencial mínima distância (DMD), como vimos no capítulo V.

Quanto à eficiência, o DSS não se mostrou supeerior ao DMD, e para ambos a probabilidade de decodificação correta tende para zero com o aumento do comprimento das sequências, à excessão da

terceira gramática (natural). Este resultado era esperado e nos mostra que a eficiência da decodificação está intimamente ligada às distancias (no sentido de Hamming) que as sequências da linguagem gerada pela fonte têm entre si. O problema da probabilidade de êrro na decodificação fica pois transferido do esquema de decodificação para as linguagens.

A vantagem do DSS sôbre o DMD está no número de "backtrackings" usados na decodificação das sequências, ou seja o tempo de decodificação. Os gráficos levantados do número médio de "backtrackings" nos mostram que além do número de "backtrackings" para o DSS ser dramaticamente menor do que para o DMD, a taxa de crescimento do primeiro (DSS) é menor que do segundo (DMD).

Uma tentativa de solucionar o problema da probabilidade de decodificação correta tender para zero com o comprimento das sequências é introduzir redundâncias nas produções. O efeito deste procedimento seria aumentar a separação das sequências da linguagem. A maneira mais eficiente de se fazer isto é um tema interessante para pesquisas, qeu resultaria na construção de linguagens adequadas ao teleprocessamento.

O modelo geral do sistema de transmissão de informação quando a fonte é gramatical, pode ser aplicado, em alguns casos, ao sistema formado por computador e seu usuário. As "mensagens" transmitidas seriam as instruções do programa de computação, o computador seria o "receptor", e o usuário a "fonte" e os erros que o usuário cometesse por descuido ou distração seriam os erros introduzidos pelo camal de transmissão. O compilador poseria fazer o "parsing" das sentenças do programa como o DSS e neste caso aceitaria instruções que contivessem erros sintáticos. Assim economizaríamos o tempo que o usuário gastaria na correção desses erros de seu programa.

Vários compiladores, mais notavelmente os compiladores para CORC (Conway e Maxwell (8), e Freeman(11)), CUPL e PL/C

(Connay et al(9)), tentam "corrigir" os erros do programa e executá-lo. Assim, todo programa é executado, independente do número de êrros que contenha.

Um compilador baseado no DSS só poderia corrigir erros de substituição. O DSS é incapaz de detetar e corrigir erros de insersão ou apagamento. A construção de decodificadores sintáticos capazes de corrigir estes tipos de erros é umaassunto que precisa ser pesquisado.

Uma idéia que talvez pudesse ser aplicada seria a de considerar dois níveis de erro. O primeiro seria o dos "êrros sintáticos", que incluiria os erros de apagamento e insersão. Isto poderia ser feito através da inclusão de produções adicionais à gramática. Para toda produção da forma $A \rightarrow aw$, onde $A \in N$, $w \in N^*$ e $a \in T$, acrescentaríamos $A \rightarrow w$ e $A \rightarrow aXw$, onde x é um novo não-terminal. Haveria também uma nova produção $X \rightarrow x$, onde x é um novo terminal. No canal de transmissão x seria levado a todos os terminais, à exceção de si próprio. A aplicação das regras $A \rightarrow w$, e $A \rightarrow aXw$ correspoderiam aos erros de apagamento e insersão respectivamente.

Um outro tópico para pesquisa futura, o mais atraente e talvez o mais difícil, é a construção de decodificadores eficientes para as linguagens naturais. No presente trabalho estudou-se uma gramática que gera frases do português. O DSS mostrou-se eficiente na decodificação de sequências da linguagem. gerada por esta gramática. Seria interessante aplicar o DSS para gramáticas com vocabulários terminais extensos e maior número de produções.

REFERÊNCIAS BIBLIOGRÁFICAS

1) Abramson, N. M., Information Theory and Coding, Mc Graw Hill Book Co., New York, 1963.

2) Ash, R., Information Theory, John Wiley and Sons, New Yoek, 1965.

3) Banerjii, R. B., "Phrase Structure Languages, Finite Machines, and Channel Capacity", Information and Control, Vol.6, 1963, pp. 153-162.

4) Beckmann, P., "Computerization of English", IEEE Spectrum, 1971, pp. 20-27.

5) Booth, T. L., "Probabilistic Representation of Formal Languages", IEEE Conf. Rec., 10th Annual Symp. on Switching and Automata Theory, Waterloo, Ontario, pp. 74-81.

6) Chomsky, N., "Three Models for the Description of Languages", IRE Trans. Inform. Theory, Vol.IT2, 1956, pp.113-124.

7) Chomsky, N., "On Certain Formal Properties of Grammars", Information and Control, Vol.2, 1959, pp. 137-167.

8) Conway, R. W., and Maxwell, W. L., "CORC- the Cornell Computing Language" CACM, junho 1963, pp. 317-321.

9) Conway, et al. "PL/C- A High Performance Subset of PL/I", TR 70-55, Computer Science, Cornll Univ., 1970.

10) Dierks, P.,"On the Distance Properties of Finite State Languages with Application to Error Correction", tese de Ph. D., Department of Electrical Engineering, Notre Dame, Indiana, maio 1971, 93p.

11) Freeman, D. Error Correction in CORC- the Cornell Computing Language, tese, Cornell Univ., 1963.

12) Gallager, R. G. Information Theory and Reliable Communication, John Wiley and Sons, Inc, New York, 1968, 588 p.

13) Ginsburg, S. and Rose, G. F., "Preservation of Languages by Transducers", Information and Control, Vol. 9, 1966, pp. 153-176.

14) Ginsburg, S. and Greibach, S., "Deterministic Contest-Free Languages", Information and Control, Vol. 9, 1966, pp. 620-648.

15) Gries, D., Compiler Construction for Digital Computers, Johon Wiley and Sons, Inc., 1971, New York, 493 p.

16) Hopcroft, J. E. and Ullman, J., Formal Languages and Their Relation to Automata, Addison-Wesley, 1969, 241 p.

17) Knuth, D. E. "Semantics of Context- Free Languages", Proc. of Symp. in Appl. Math., Vol XIX, American Mathematical Society, Providence, R. I., 1967, pp. 52-110.

18) Martin, J. Telecommunications and the Computer, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1967, 317 p.

19) Minsky, M. L. Computation: Finite and Infinite Machines,

Prentice-Hall, Imc., Englewood Cliffs, N. J. , 1967, 317 p.

20) Pollack, B. W. Compiler Techniques, Auerbach Publishers, Inc., 1972, 692 p.

21) Rosenkrantz, D. J., "Programmed Grammars - A New Device for Generating Formal Languages", IEEE Conf. Rec., 8th Annual Symp. on Switching and Automata Theory, 1967, pp. 14-20.

22) Salomaa, A. "Probabilistic and Weighted Grammars", Information and Control, Vol. 15, 1969, pp. 529-544.

23) Shannon, C. E. "A Mathematical Theory of Communications", Bell System Tech. Journal, Vol. 27, 1948, pp. 379-423, 623-656.

24) Souza, C. R. and Sholtz, R. A. "Syntactical Decoders and Backtracking S-Grammars", The Aloha System Techn. Report A69-9, Univ. of Hawaii, Honululu, novembro 1969, 33 p.

25) Souza, C. R. "Probabilities in Context- Free Programmed Grammars", The Aloha System Techn. Report A70-2, Univ. of Hawaii, Honululu, março de 1970, 48 p.

26) Souza, C. R. and Sholtz, R. A. "Probabilistic Generation, Transmission and Syntactical Decoding of Context- Free Languages", Proc of the 8th Allerton Conference on Circuit and System Theory, Monticello, Illinois, outubro 1970.

27) Wozencraft and Jacobs Principles of Communication Engineering, John Wiley and Sons, Inc., New York, 1965, 720 p.

APÊNDICE A

Demonstração do teorema 4.2

Prova parte a

a.1 Prova a.1 verificando para $i < 1$ que a visita mais recente de qualquer antecedente de x deve ser uma F-visita, pois x só pode ser atingido de nó antecedente por meio de um movimento para frente (regra 1 ou 2). Para $i = 1$, $z_1 \leq T_1$ pois o ponteiro se encontra no nó x e tal só é possível se o valor de x não violar o limiar corrente.

a.2 e a.3 A prova desses dois itens será feita por indução nos sucessivos nós visitados pelo ponteiro.

A base da indução é o nó origem ($l = 0$), o qual satisfaz trivialmente a.2 e a.3 pois não existe i tal que $0 \leq i \leq -1$.

Passo da indução: seja a.2 e a.3 satisfeitas para o nó x, com valores do caminho $z_0, z_1, \ldots z_l$ e limiares do caminho $T_0, T_1, \ldots T_l$. Temos então duas possibilidades: o nó que está sendo olhado é um descendente de x ou é o antecedente de x. Chamemos a primeira possibilidade de "estado A", ou seja, desigualdades satisfeitas, ponteiro olhando para frente e movimento não concretizado. A segunda possibilidade será o "estado B" que consiste em: desigualdades satisfeitas, ponteiro olhando para trás e movimento não concretizado.

Demonstraremos que o ponteiro até se movimentar estará sempre ou no estado A ou B, e quando se movimentar, as desigualdades continuarão a serem satisfeitas.

No estado A, temos tres casos possíveis: $z_{l+1} \leq T_l$ e $z_l > T_l - \Delta$, $z_{l+1} \leq T_l$ e $z_l \leq T_l - \Delta$ e finalmente $z_{l+1} > T_l$.

No último caso, pela regra 3, vemos que o ponteiro irá para o estado B. Nos dois primeiros casos o ponteiro se movimentará para frente e o limiar será no máximo abaixado, satisfazendo a.2. Se $T_{l+1} < T_l - \Delta$, ou seja, o limiar foi abaixado, então foi aplicada a regra 1 e neste caso $z_l > T_l - \Delta$, satisfazendo a.3.

No estado B pode acontecer dois casos: o da regra 4, $z_{l-1} \leq T_l$ ou da regra 5, $z_{l-1} > T_l$. No caso da regra 4, o ponteiro se movimentará para trás, não se alterando os limiares , satisfazendo as inequações a.2 e a.3. Além disso, como $z_{l-1} \leq T_l$ e $T_l \leq T_{l-1} - \Delta \Rightarrow T_l < z_{l-1}$, então devemos ter $T_l > T_{l-1} - \Delta \Rightarrow T_l = T_{l-1}$ (a.5) pois por a.1 $T_l \quad T_{l-1}$.

A equação a.5 nos será útil na demonstração da parte b do teorema. Se por outro lado, $z_{l-1} > T_l$, o movimento não se concretizará. Como por a.1 $T_{l-1} \geq z_{l-1}$, temos que $T_{l-1} > T_l$, ou seja, $T_{l-1} \geq T_l + \Delta$. O limiar T_l é pois alterado mas a.2 continua valendo. O mesmo acontece com a.3 pois não mudamos T_{l-1}. Pela regra 5, o ponteiro irá olhar para frente, indo ao estado A.

Provamos então o seguinte: se o ponteiro se movimentar a.2 e a.3 continuarão a valer e se por acaso o ponteiro não se movimentar passará do estado A para B ou vice versa e as desigualdades continuarão valendo.

Para demonstrar a parteb, é necessário provar primeiro o seguinte corolário de a:

Corolário Se x é F-visitado com limiar final T, então T é o limiar inicial das F-visitas subsequentes aos descenden-

tes imediatos de x para os quais $z \leqslant T$.

Prova: Seja y descendente imediato de x e $z_x \leqslant T$. Se y é o melhor nó descendente, então y será alcançado logo após a F-visita (regra 1 ou 2) e o corolário é estabelecido. Se y não é o melhor nó, então este é alcançado depois de x ter sido alcançado através de um movimento para trás (regra 4). Como por a.5, $T_{l+1} = T_l = T$, temos que o limiar inicial para o nó y é T.

a.4) a.4 é uma consequência direta de a.3 e de a.1.

b) Para se demonstrar o item b, é necessário usar (indução) no comprimento dos nós, verificando primeiro o teorema para a origem e depois assumindo válido para um dado nó, demonstrar que é válido para seus descendentes imediatos.

base: Para a origem o teorema é satisfeito trivialmente, pois a única F-visita se dá no início da decodificação, e como $T_0 = z_0$, a desigualdade é válida.

passo: Seja então válido o teorema para o nó x, onde $|x| = l-1$, então o limiar final T da primeira F-visita obedece a

$T-\Delta \leqslant z_{l-1} < T$, onde z_{l-1} é o valor de x.

Seja y descendente imediato de x com valor z_l. Temos duas possibilidades:

a- $z_l \leqslant T$

b- $z_l > T$

a- Se $z_l \leqslant T$, a desigualdade é satisfeita (regra 1 é aplicada) pois para todo descendente imediato de x, T é o limiar inicial das F-visitas a estes descendentes (como vimos pelo corolário).
Se $T' = T$, onde T' é o limiar mais baixo tal que $T' \geqslant z'$ e z' é o valor do antecedente de x, então o limiar

da segunda F-visita a y é o mesmo que o limiar final da segunda F-visita a x. Isto acontece porque o ponteiro ao chegar a x por um movimento para trás, e esgotados os nós descendentes imediatos com valor menor que T, pela regra 3 o ponteiro irá para o nó antecedente. Assim sendo, a próxima visita a x será uma F-visita com limiar final igual a $T+\Delta$. Nas visitas subsequentes, quando o ponteiro F-visitar x, F-visitará y com o mesmo limiar final (corolário e regra 2). Se $T' > T$, então a segunda F-visita ao nó descendente de x será logo após o ponteiro ter atingido x e tentado ir para trás (regra 5). Então neste caso o limiar é aumentado de e o ponteiro fará um movimento para frente pela regra 2. O limiar final da segunda F-visita a y será Δ acima de T, ou seja, aumentado de $n\Delta$ em relaão à primeira visita. Após isto, toda vez que o ponteiro tentar ir para o antecessor de x, aumentará o limiar de Δ e F-visitará y. Quando o antecessor de x puder ser alcançado, será possível a segunda F-visita a x e daí em diante, por hipótese o limiar só será aumentado de Δ .

b- Se z_1 T, temos também duas possibilidades:

b.1 $z_1 \leqslant T' - \Delta$

b.2 $z_1 > T' - \Delta$, onde T' é o segundo limiar final de x.

No caso b.2, y não será atingido antes da segunda F-visita a x, pois $T' - \Delta \geqslant z'$ por hipótese. Em outras palavras, o ponteiro irá primeiro ao antecessor de x antes de ir ao seu sucessor. Depois desta visita, o limiar só será aumentado de em cada F-visita subsequente a x, o que nos prova a desigualdade e a condição adicional.

Em b.1, temos qeu o ponteiro só alcançará o antecessor de x em um movimento para trás quando o limiar for igual a $T' - \Delta$ (por a.5). Até isto acontecer, o limiar é aumentado cada vez de e tenta o movimento para frente. Pelo corolário vemos que a desigualdade e a condição adicional são também satisfeitas.

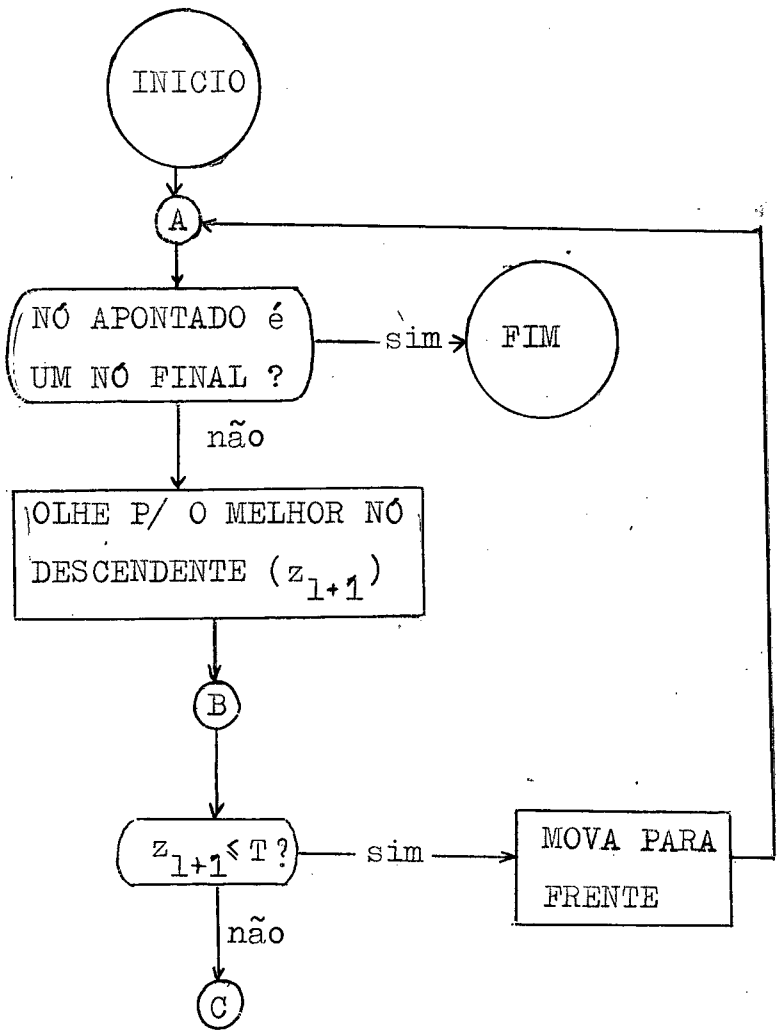
c) Seja o nó x onde $|x| = 1$, F-visitado com limiar final T_1. Queremos mostrar que todo descendente de x para o qual o limiar está abaixo de T_1 deve ser F-visita com limiar final T_1.

Na prova da parte b mostraremos que cada descendente imediato de x para o qual $z_{1+1} \leqslant T_1$ é F-visitado com limiar final T_1. Por indução, temos que o descendente y de x, para o qual $|y| = 1+j$ é F-visitado com limiar final T_1 antes que y seja novamente visitado.

Por a.2 tiramos a segunda parte do item c, ou seja, que o limiar não será aumentado acima de T até que x seja novamente F-visitado.

APÊNDICE B

O funcionamento do algoritmo de decodificação é descrito pelos diagramas de blocos dados abaixo.



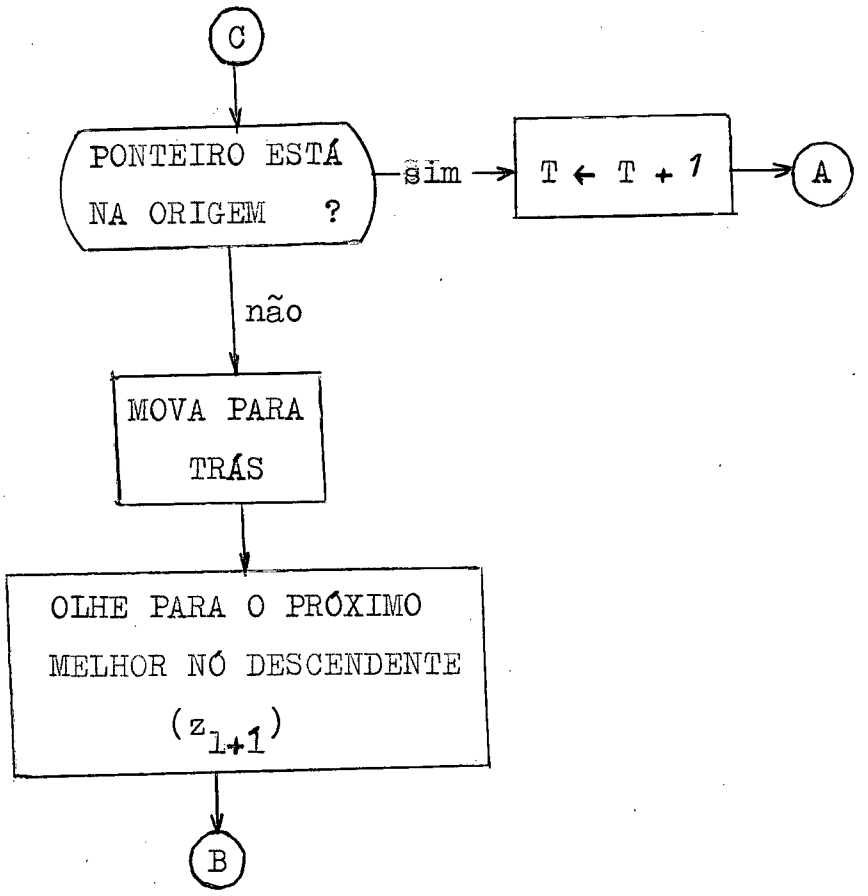


DIAGRAMA DE BLOCOS PARA O DMD

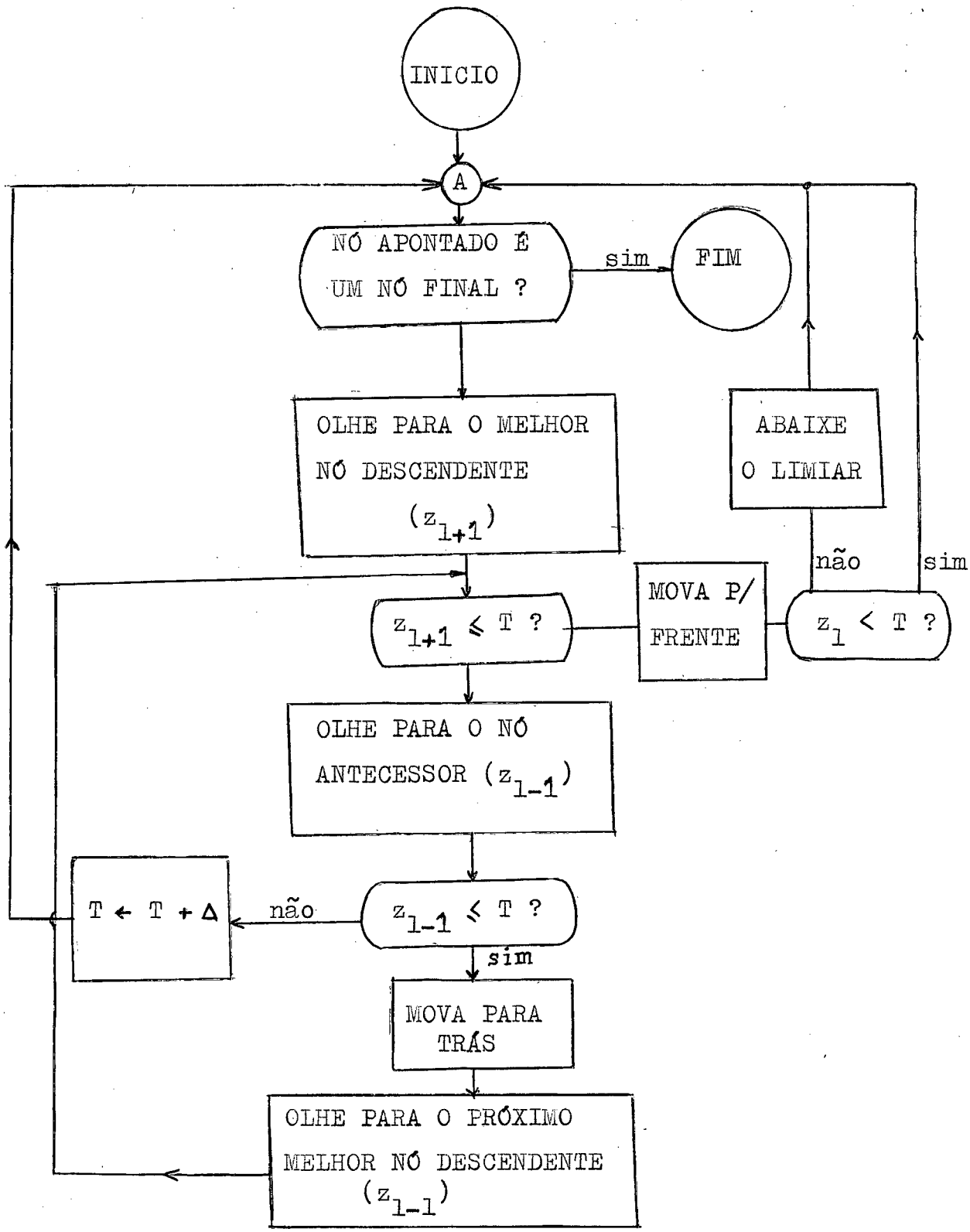


DIAGRAMA DE BLOCOS PARA O DSS

A implementação do algoritmo foi feita no computador 1370 da IBM. Para cada simulação do algoritmo foi tomada uma amostra diferente de sequências da linguagem. Para a primeiragramática, ou seja, a que gera sentenças "especulares" num vocabulário bi - nário, tomou-se 100 sequências para cada comprimento. No caso da segunda gramática, a do cálculo proposicional, o número de sequências para cada comprimento foi variável, com um máximo de 100 e um número total de 1300. Para a terceira gramática, a que gera frases do português, a amostra tomada foi de 100 sequências da linguagem para cada diferente. As sequências consideradas tinham no máximo 200 símbolos (letras). Para esta gramática consideramos todas as palavras como de mesmo comprimento (=7), completando com brancos os símbolos que faltavam.

As figuras b.1,b.2,b.3,b.4,b.5,b.6 mostram saídas típicas dos programas.

As figuras b.1 e b.2 mostram o DSS e o DMD para a primeira gramática e as figuras b.3 e b.4 o DSS e o DMD para a segunda gramática.

Nas figuras "FRASE" é a sequência gerada pela fonte,"SEQUE" é a sequência após a passagem pelo canal de transmissão e "SAIDA" é a sequência decodificada, "CONT"nos dá o número de "backtrackings" usados na decodificação das sequências e "HORA" o tempo em hora, minutos, segundos e milisegundos a cada passo do programa.

Nos casos em que o comprimento da FRASE é maior que 19 não temos a SAIDA.

As figuras b.5 e b.6 mostram o DSS e o DMD para a segunda gramática. "FRA" é a sequência gerada pela fonte, "SEQ" é esta sequência depois de passar pelo canal de transmissão e "LDT" é a sequência decodificada. "CONT" e "HORA" têm o mesmo significado que nas duas primeiras gramáticas.

```
SEQUE='010101101C0001C10011 0C101010';                    HORA='043239010';
SEQUE='010101101C0001C10011 0C101010';
SAIDA='010101101100001C10001 101101010'                   CONT=      1027
LG=     29;

FRASE='110110101C0101011011';                             HORA='043239940';
SEQUE='110110101C01C1C11011';                             HORA='043240010';
SEQUE='110110101C01C1C11011';
SAIDA='110110101C0101011011'                              CONT=       138
LG=     21;

FRASE='1011OC01101';      HORA='043240370';
SEQUE='1011OC01111';      HORA='043240390';
SEQUE='1011OC01111';
SAIDA='1011OC01101'    2  CONT=    3     9      4         D=  5       1;    6
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8
FRASE='1011100110C01100111011';                           HORA='043240540';
SEQUE='C0111001C0C0C1001110 1';                           HORA='043240570';
SEQUE='C0111001C0C0C1001110 1';
SAIDA='1011100100C0010011101'                             CONT=       436
LG=     21;

FRASE='1111C1111';        HORA='043240990';
SEQUE='C111C1111';        HORA='043241010';
SEQUE='C111C1111';
SAIDA='1111C1111'         CONT=        10                  D=          1;

FRASE='0C0';              HORA='043241090';
SEQUE='0C1';              HORA='043241110';
SEQUE='0C1';
SAIDA='0C0'               CONT=         2                 D=          1;

FRASE='10010 11C1101001';HORA='043241270';
SEQUE='C0010CCC1C01001';HORA='043241310';
SEQUE='C0010CCC1C01001';
SAIDA='100100 1C1001001'  CONT=       515                 D=          1;

FRASE='00C00';            HORA='043241920';
SEQUE='00C0C';            HORA='043241940';
SEQUE='00C0C';
SAIDA='00C00'             CONT=         4                 D=          1;
```

figura b.1 (DMD)

```
0          1          2          3          4          5          6
FRASE='1C1';          HORA='205611680';
SEQUE='1C1';          HORA='205611710';
SAIDA='1C1'           CONT=       0          D=      275;

FRASE='1000101011010C010110101000 1';          HORA='205611780';
SEQUE='1000101011010C110110C110011';           HORA='205611820';
SAIDA='1000101011010C010110101000 1'           CONT=       0
LG=     27;

FRASE='0110111001C100111011 0';                HORA='205611930';
SEQUE='01101C1001C1001110C00';                 HORA='205611970';
SAIDA='0110111001C100111011 0'                 CONT=      14
LG=     21;

FRASE='1100101101010110101000 1001001 01';     HORA='205612100';
SEQUE='1100101101010110101000 1C01001 01';     HORA='205612150';

FRASE='0100101010010 1C10100101010010';        HORA='205612200';
SEQUE='0100101010010 1CC0100101010010';        HORA='205612240';
SAIDA='0100101010010 1C10100101010010'         CONT=       0
LG=     29;

FRASE='1C1';          HORA='205612380';
SEQUE='1C1';          HORA='205612400';
SAIDA='1C1'           CONT=       0          D=      280;

FRASE='0001001010101011C1101010100 10';        HORA='205612480';
SEQUE='0001001010101011C1101110C0100 10';      HORA='205612540';

FRASE='0110101 0C01010110';                    HORA='205612560';
SEQUE='01C0101 0C01010C10';                    HORA='205612590';
SAIDA='0110101 0C01010110'                     CONT=       3
LG=     17;
0          1          2          3          4          5          6
FRASE='0C0';          HORA='205612680';
SEQUE='0C0';          HORA='205612700';
SAIDA='0C0'           CONT=       0          D=      282;

FRASE='01C10';        HORA='205612750';
SEQUE='01C10';        HORA='205612760';
SAIDA='01C10'         CONT=       0          D=      283;

FRASE='0C0';          HORA='205612830';
SEQUE='0C0';          HORA='205612860';
SAIDA='0C0'           CONT=       0          D=      284;
```

figura b.2

(DSS; Δ = 3.0)

```
FRASE='NY';                HORA='092849140';
SEQUE='EY';                HORA='092849160';
SEQUE='EY';
SAIDA='NY'                 CONT=        0          D=        1;

FRASE='(YE(((XU(NYENY))U(((NNYENY)U(YE';          HORA='092849240';
SEQUE='(YE(((XU(EYENY))U(((NXYENY)U(YE';          HORA='092849280';
SEQUE='(YE(((XU(EYENY))U(((NXYENY)U(YE';

FRASE='(XU(XUNX))';        HORA='092849320';
SEQUE='(XU(XUXX))';        HORA='092849340';
SEQUE='(XU(XUXX))';
SAIDA='(XU(XUNX))'         CONT=       14          D=        1;

FRASE='(XUNY)';            HORA='092849410';
SEQUE='(XUXY)';            HORA='092849420';
SEQUE='(XUXY)';
SAIDA='(XUNY)'             CONT=        8          D=        1;

FRASE='((XUY)ENNY)';       HORA='092849470';
SEQUE='((XUY)ENXE)';       HORA='092849490';
SEQUE='((XUY)ENXE)';
SAIDA='((XUY)ENNX)'        CONT=      280          D=        1;

FRASE='(NYUY)';            HORA='092849850';
SEQUE='(UYUY)';            HORA='092849870';
SEQUE='(UYUY)';
SAIDA='(NYUY)'             CONT=        2          D=        1;

FRASE='(XU((N((XU((XU(N(((YENY)EN(YE(Y';          HORA='092849950';
SEQUE='(XX(((X(((XY((XU(N(((YEUY)EE(NE(Y';        HORA='092850000';
SEQUE='(XX(((X(((XY((XU(N(((YEUY)EE(NE(Y';

FRASE='(YU((XEY)EX))';     HORA='092850040';
SEQUE='(YU((XEN)EX))';     HORA='092850060';
SEQUE='(YU((XEN)EX))';
SAIDA='(YU((XEX)EX))'      CONT=       69          D=        1;

FRASE='((N(YE(YE(XU(NN((NYU(NXU(XU(N((';          HORA='092850210';
SEQUE='((N(YE(YE(XU(UU((XYU(EXU(XN(E((';          HORA='092850250';
SEQUE='((N(YE(YE(XU(UU((XYU(EXU(XN(E((';

FRASE='((YEX)EX)';         HORA='092850290';
SEQUE='((YEX)EXE';         HORA='092850300';
SEQUE='((YEX)EXE';
SAIDA='((YEX)EX)'          CONT=       62          D=        1;

FRASE='(NNYUX)';           HORA='092850420';
SEQUE='(NU)UX)';           HORA='092850440';
```

figura b.3 (DMD)

```
SEQUE='(XUY)';                HORA='200253120';
SAIDA='(XUY)'                 CONT=      0          D=      271;

FRASE='((N(N(XEX)EX)E(N(XE(NYE(XEX)))E';          HORA='200253190';
SEQUE='((N(N(XEX)EX)E(N(XX(XYE(XEX)))E';          HORA='200253240';

FRASE='(YE(YE((XUX)E(((YENY)UY)E(YENY)';          HORA='200253290';
SEQUE='(YE(YE((X)X)E(((YENY)UY)E()ENN)';          HORA='200253340';

FRASE='(XEX)';                HORA='200253350';
SEQUE='(XEX)';                HORA='200253370';
SAIDA='(XEX)'                 CONT=      2          D=      272;

FRASE='(XUY)';                HORA='200253420';
SEQUE='(XUY)';                HORA='200253430';
SAIDA='(XUY)'                 CONT=      0          D=      273;

FRASE='NY';                   HORA='200253470';
SEQUE='N)';                   HORA='200253480';
SAIDA='NY'                    CONT=      6          D=      274;

FRASE='(N(YEX)UY)';           HORA='200253550';
SEQUE='(N(YEX)UY)';           HORA='200253570';
SAIDA='(N(YEX)UY)'            CONT=      0          D=      275;

FRASE='NN(YEY)';              HORA='200253620';
SEQUE='UX(YEU)';              HORA='200253630';
SAIDA='NN(YEX)'               CONT=    147          D=      276;

FRASE='(((((XUY)EX)U(XUX))UY)UY)EX)';             HORA='200254700';
SEQUE='(((((XUY)EX))(XUX)(UN)UY)EX)';             HORA='200254750';
SAIDA='(((((XUY)UX)E(XUX))UY)EY)UX)'              CONT=     26
LG=      29;

FRASE='(XU(N(N(NYEY)U(XUNY))UN(XU(YUY)';          HORA='200255130';
SEQUE='(XU(N(N(N(EE)U(XNEY))UX(XU(Y)))';          HORA='200255180';

FRASE='(N(YEX)U(YEX))';       HORA='200255210';
SEQUE='(N()EX)U(NEX))';       HORA='200255300';
SAIDA='(N(YEX)U(YEX))'        CONT=     37          D=      278;

FRASE='(NYU((YENX)UY))';      HORA='200255680';
SEQUE='(NEU((EUNX)UE))';      HORA='200255700';
SAIDA='(NYU((XUNX)EX))'       CONT=     89          D=      279;

FRASE='NY';                   HORA='200256430';
SEQUE='UU';                   HORA='200256440';
SAIDA='NX'                    CONT=    356          D=      280;
```

figura b.4 (DSS, $\Delta = 2.0$)

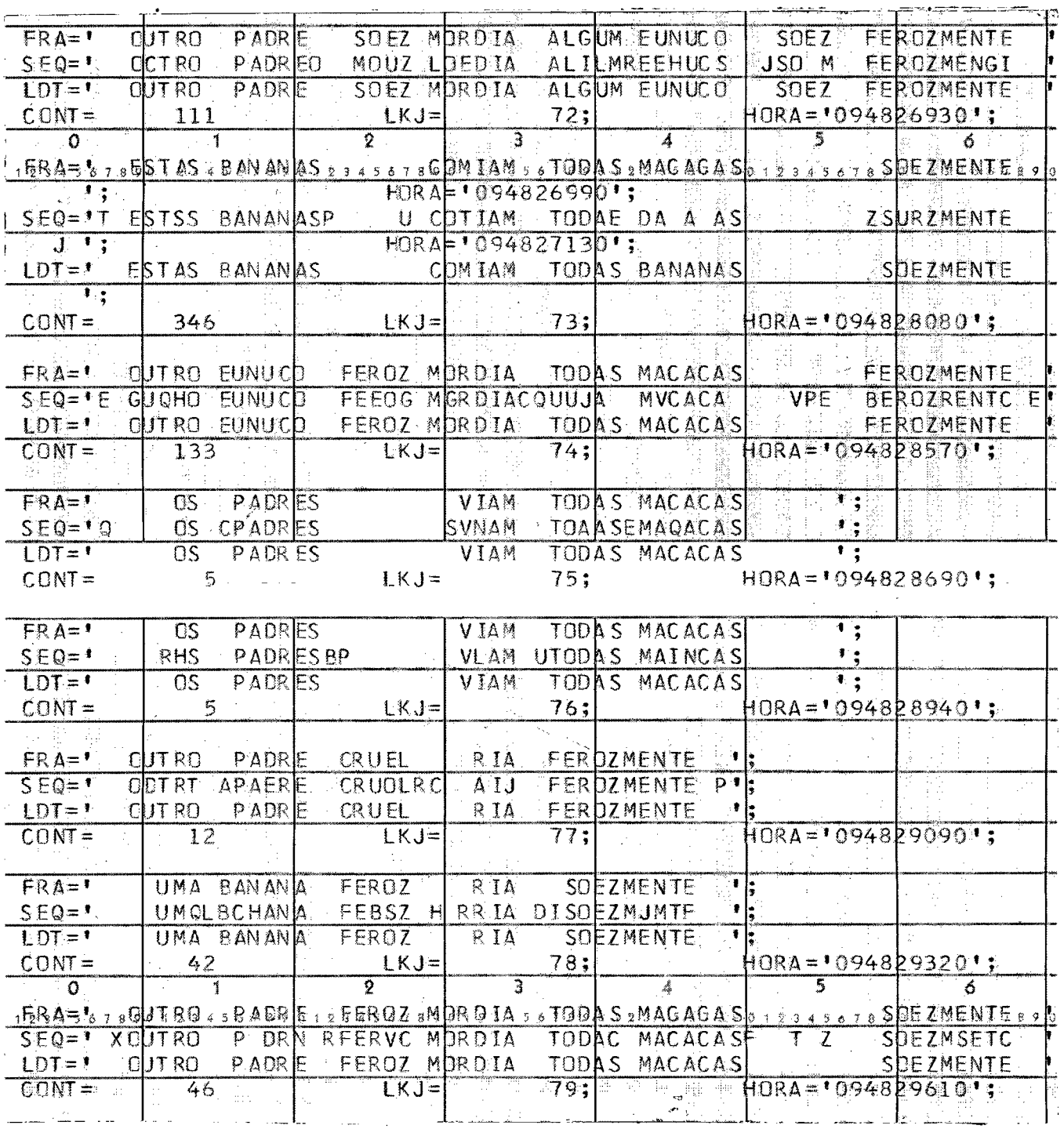


figura b.5 (DMD, = 4.0)

FRA='	UMA MACACA	SOEZ MORDIA	TODAS BANANAS			CRUELMENTE

SEQ='	UTM MAEAZAV	SCXU MOODI O	TLDAS BPZANAS		O	FCRUELMENNEA
LDT='	UMA MACACA	SOEZ MORDIA	TODAS BANANAS			CRUELMENTE
CONT=	0	LKJ=	49;		HORA='105234850';	
FRA='	UMA MACACA	SOEZ	RIA	CRUELMENTE	';	
SEQ=' F	DUMA MACATA	SOEZ	RIA	TRUGLMENTE	';	
LDT='	UMA MACACA	SOEZ	RIA	CRUELMENTE	';	
CONT=	0	LKJ=	50;		HORA='105235100';	
FRA='	UMA BANANA	SOEZ	RIA	CRUELMENTE	';	
SEQ='UJ	DMAHBBNANA	SO Z D	IRIA	LCRUELMENTE	';	
LDT='	UMA BANANA	SOEZ	RIA	CRUELMENTE	';	
CONT=	0	LKJ=	51;		HORA='105235250';	
FRA='	OS PADRES		COMIAM	TODAS BANANAS		CRUELMENTE
SEQ='	OS PXDRES		OMIAM	TGDIH BHNANAS		CRUELMERBG
LDT='	OS PADRES		COMIAM	TODAS BANANAS		CRUELMENTE
CONT=	0	LKJ=	52;		HORA='105235500';	
FRA='	UMA BANANA	CRUEL	RIA	FEROZMENTE	';	
SEQ='N	HUMA BANAAA	LRUELEB	MRIA	FEROZEONTE	';	
LDT='	UMA BANANA	CRUEL	RIA	FEROZMENTE	';	
CONT=	0	LKJ=	53;		HORA='105235740';	
FRA='	OUTRO PADRE	FEROZ	RIA	CRUELMENTE	';	
SEQ='	GETLD UPADRE	FEROZ	RIA	XHUELBENTE Q	';	
LDT='	OUTRO PADRE	FEROZ	RIA	CRUELMENTE	';	
CONT=	0	LKJ=	54;		HORA='105235990';	
FRA='	OUTRO EUNUCO	CRUEL MORDIA	ALGUM	PADRE	CRUEL	FEROZMENTE
SEQ='	OUTRO PULUCO	CRUEL MORDIA	LGAM	PADJE	CLUEL	FEUAZHENTE
LDT='	OUTRO EUNUCO	CRUEL MORDIA	ALGUM	PADRE	CRUEL	FEROZMENTE
CONT=	0	LKJ=	55;		HORA='105236300';	
FRA='	UMA MACACA	SOEZ MORDIA	TODAS BANANAS			SOEZMENTE
SEQ='	CUMA MUCACAH	SOCZ MORDIE	LTCIJJ QANRNAS		M U Q E	SONBMENTG
LDT='	UMA MACACA	SOEZ MORDIA	TODAS BANANAS			SOEZMENTE
CONT=	0	LKJ=	56;		HORA='105236550';	

figura b.6 (DSS, Δ = 4.0)