1. Publicação nº	2. Versão	3. Data	5. Distribuição
INPE-2916-PRE/428		*Out., 1983*	☐ Interna ☒ Externa ☐ Restrita

4. Origem	Programa
DMC/DGC	*TEREAL*

6. Palavras chaves - selecionadas pelo(s) autor(es)
ÁRVORES BINÁRIAS
NÚMERO MÉDIO DE ACESSO
OTIMIZAÇÃO

7. C.D.U.: *681.3.016*

8. Título *INPE-2916-PRE/428*

ÁRVORES BINÁRIAS COM ENDEREÇOS ORDENADOS: UMA TÉCNICA PARA OTIMIZAÇÃO DE ACESSO PARA ÁRVORES EM DISCO.

10. Páginas: *13*

11. Última página: *11*

12. Revisada por

Edson Luis França Senne

9. Autoria *Samuel Alves Pereira*
Eliane D'Ippolito
Hans Jurgen Seehusen

13. Autorizada por

Nelson de Jesus Parada
Diretor Geral

Assinatura responsável

14. Resumo/Notas

Árvores binárias são bons métodos para organização de arquivos dinâmicos. Se uma árvore binária for armazenada numa memória de acesso pseudo-aleatório como disco, pode-se usar a técnica de Árvores Binárias com Endereços Ordenados (ABEO) para diminuir o número médio de acessos e transferências. São apresentados os algoritmos de inserção e eliminação para manipulações de ABEO. Os resultados, obtidos por simulação, mostram que o número médio de acessos a uma ABEO em disco pode ser reduzido até a metade em relação a árvores binárias comuns. Para manter o balanceamento, foi usada a técnica de árvores virtuais.

15. Observações

Este trabalho foi apresentado no 10º Seminário Integrado de "Software" e "Hardware", Campinas, 25 a 29 de julho de 1983.

ÁRVORES BINÁRIAS COM ENDEREÇOS ORDENADOS: UMA TÉCNICA PARA OTIMIZAÇÃO DE ACESSO PARA ÁRVORES EM DISCO.

PEREIRA, S.A.*; D'IPPOLITO, E.**; SEEHUSEN, H.J.***

SUMÁRIO

Árvores binárias são bons métodos para organização de arquivos dinâmicos. Se uma árvore binária for armazenada numa memória de acesso pseudo-aleatório como disco, pode-se usar a técnica de Árvores Binárias com Endereços Ordenados (ABEO) para diminuir o número médio de acessos e transferências. São apresentados os algoritmos de inserção e eliminação para manipulações de ABEO. Os resultados, obtidos por simulação, mostram que o número médio de acessos a uma ABEO em disco pode ser reduzido até a metade em relação a árvores binárias comuns. Para manter o balanceamento, foi usada a técnica de árvores virtuais.

ABSTRACT

Binary trees are good methods for dynamic file organization. If a binary tree is stored on a pseudo random access store as a disk, it is possible to use the Ordered Address Binary Trees (OABT) technique for optimizing the mean number of accesses and transfers. The insertion and deletion algorithms to manipulate the OABT technique are presented. Simulation results have shown that the mean access number to OABT on disk can be reduced down to its half, compared to common binary trees. The virtual tree technique was used to maintain the balancing.

* Mestre em Ciência (ITA, 1981); Banco de Dados, Organização de Arquivos, "Software" de Controle, Linguagens de Programação e Análise Numérica; Assistente de Pesquisa (DGC-DMC), Instituto de Pesquisas Espaciais (INPE), S.J.Campos, SP, 12200.

** Tecnóloga em Computação (ITA, 1977); Banco de Dados, Organização de Arquivos, Compiladores; Analista de Sistemas, IEAv, Centro Técnico Aeroespacial (CTA), S.J.Campos, SP, 12200.

*** Ph.D. em Informática (Univ.Tec.de Berlin, 1976); Banco de Dados, Organização de Arquivos, Inteligência Artificial, Linguagens de Programação; Pesquisador Adjunto, Chefe do Grupo "Sistemas Científicos" do Centro de Processamento de Dados do IEAv, Centro Técnico Aeroespacial (CTA), S.J.Campos, SP, 12200.

1 - INTRODUÇÃO

Apesar do grande aumento da capacidade de processamento e armazenamento dos computadores, é muito importante que os dados estejam bem organizados nos equipamentos de memória. O grande problema é armazenar e manipular arquivos com altas taxas de atualização. Um tipo de estrutura de dados apropriado para isto é a organização em árvores.

Se o acesso ao endereço físico, onde as chaves serão armazenadas, for possível, podem-se construir árvores binárias onde todos os caminhos de procura sejam ordenados com respeito a estes endereços físicos. A técnica de Árvores Binárias de Endereços Ordenados (ABEO) faz com que o tempo médio de acesso seja reduzido.

Serão apresentados nas Seções seguintes os algoritmos de uma ABEO, mostrando-se também como a taxa média de transferência é diminuída.

1.1 - Definições

Uma árvore binária A consiste em um conjunto finito de nós que é vazio ou consiste em uma raiz r e duas árvores binárias disjuntas $A_e(r)$ e $A_d(r)$, chamadas subárvores esquerda e direita da raiz r, respectivamente. As raizes de $A_e(r)$ e $A_d(r)$ são chamadas sucessoras de r. O número de sucessores de um nó x é chamado o grau de x. Um nó de grau zero é chamado folha.

Uma sequência de nós $(x_1, x_2, \ldots, x_{i+1})$ é um caminho de comprimento i, onde x_{j+1} é sucessor de x_j, $j = 1, \ldots i$. O comprimento do caminho de r até um nó x é chamado nível de x. A raiz r tem nível zero.

1.2 - Árvores Binárias com Chaves

Um registro de dados é o conjunto de todas as informações a respeito de um dado do arquivo de dados que deve ser organizado, neste caso, em árvores binárias. Cada registro é armazenado em um nó, que é identificado univocamente pelo dado, ou por parte dele, denominado chave.

Uma árvore binária com chaves é construída segundo as seguintes regras, onde $A = \{x_1, x_2, \ldots, x_n\}$ é o conjunto de nós a ser estruturado e C(x) é a chave do nó x:

$$\begin{aligned} &\text{(a) } r \in A \text{ é a raiz da árvore, escolhida;} \\ &\text{(b) } x_j \in A_e(r) \text{ se } C(x_j) < C(r); \\ &\text{(c) } x_j \in A_d(r) \text{ se } C(x_j) > C(r). \end{aligned} \tag{1.1}$$

A técnica de procura que usa a estrutura de árvores binárias com chaves chama-se procura binária.

A procura de uma chave C(x) numa árvore A é feita da seguinte maneira: PROCURA (C(x), A):

1. Se A for vazia então a procura termina sem sucesso;
2. Se C(r) = C(x) então a chave foi encontrada;
3. Se C(r) > C(x) então PROCURA (C(x), $A_e(r)$);
4. Se C(r) < C(x) então PROCURA (C(x), $A_d(r)$).

1.2.1 - Algoritmo de Inserção Simples

Para a inserção de uma nova chave C(x), deve-se executar o algoritmo de procura para C(x) acima. Como a procura termina sem sucesso, a nova chave é inserida como folha. Assim, o Passo 1 é modificado para:

1. Se A for vazia então A é substituída por x.

Esse algoritmo será referenciado pelo nome de Inserção Simples. O caminho processado durante a procura é chamado caminho de procura.

Exemplo 1 - Construir uma árvore binária com as chaves: (500, 200, 800, 900, 350), usando o algoritmo de Inserção Simples e representando a árvore por meio de um grafo direcionado.

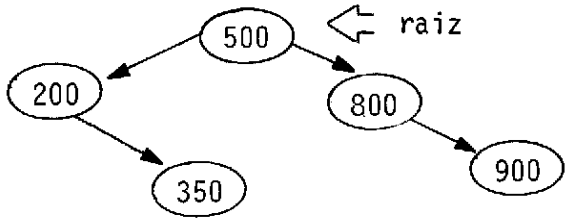

Fig. 1.1 - Árvore binária com chaves.

1.2.2 - Algoritmo de Eliminação Simples

Seja x o nó a ser eliminado. Se x for de grau zero, nenhuma mudança é feita; se o grau for 1, então x é substituído por seu sucessor. Se o grau for 2, realizar as seguintes operações:

1. Substituir x por um de seus sucessores (por exemplo a raiz da subárvore esquerda (*);
2. Colocar a outra subárvore como sucessora do nó que contiver a maior chave da subárvore esquerda.

(*) *Se for escolhida a raiz da subárvore direita, o procedimento é inteiramente simétrico.*

3. $Pd(x) \leftrightarrow Pd(y_i)$ - troca dos ponteiros que apontam para as raízes das subárvores direitas de x e de y_i.

Algoritmo INS(x)

Passo 1: Inicialização: Pe(x): = Pd(x): = "."; C(x): = nova chave; i = 0;
Passo 2: Examinar CP(x): i: = 1+1; Se i > k então x é sucessor de y_k e fim do processo;
Passo 3: Verificar a necessidade de troca: Se $E(C(x)) > E(C(y_i))$ ir para o passo 2;
Passo 4: Realização da troca: Troca (x, y_i) ir para o passo 2.

Se o passo 4 for omitido (troca de nós), o algoritmo resultante é o de inserção simples da Seção 1.

O algoritmo INS pode ser adaptado para usar a técnica de Árvores Virtuais na realização do balanceamento da árvore. Esta adaptação é simples pelo fato de ser possível mudar uma chave de um lugar para outro. O algoritmo, de forma recursiva, é:

INS(x, r);
1. Balanceamento: Se $|C(x) - C'(r)| < |c(r) - C'(r)|$ então $C(x) \leftrightarrow C(r)$;
2. Ordenação dos Endereços: Se $E(C(x)) < E(C(r))$ então troca (x, r);
3. Andar no Caminho de Inserção: Se CP(x) já tiver sido percorrido, então inserir x como folha e terminar o processo senão: Se C(r) < C(r), então INS(x, Ae(r)), senão: INS(x, Ad(r)).

2.2 - Algoritmo de Eliminação - (EL)

Com a possibilidade de mudar a informação de um lugar para outro, o algoritmo de eliminação poder ser descrito recursivamente como abaixo, onde MA(x) e ME(x) são os nós que contêm a maior e a menor chave da subárvore esquerda e direita de x, respectivamente.

Algoritmo EL(x)

Passo 1: Se x for folha, desconectá-la da árvore e terminar o algoritmo;
Passo 2: Procurar M(x) = MA(x) ou M(x) = ME(x)(*);
Passo 3: C(x): = C(M(x));
Passo 4: EL(M(x)).

(*) *A escolha de MA(x) ou ME(x), para x de grau 2, não afeta a eficiência de acesso da árvore [3]. Se x for de grau 1, deve-se procurar o nó correspondente à subárvore existente.*

1.3 - Tempo Médio de Acesso em uma Árvore Binária com Chaves

Supondo que cada sequência de inserção tenha a mesma probabilidade de ocorrer e que as chaves sejam procuradas com a mesma frequência, calcula-se o tempo médio de acesso ($\overline{T}$) a uma árvore binária com chaves por:

$$\overline{T} = (1/n!) \sum_{k=1}^{n!} \; (1/n) \sum_{j=1}^{n} \; \sum_{i=1}^{|cp(x_j)|} t(x_i, x_{i+1}) \;, \qquad (1.2)$$

$\sum_{i=1}^{|cp(x_j)|}$: um caminho

$\sum_{j=1}^{n}$: uma árvore

$\sum_{k=1}^{n!}$: todas as sequências de inserção

onde n = número de nós da árvore;
$|cp(x)|$ é o comprimento do caminho para o nó x;
$t(x, y)$ é o tempo para ir do nó x até y.

Se o tempo médio de deslocamento de um nó até o seu sucessor for constante e igual a c', a equação (1.2) pode ser aproximada por:

$$\overline{Ac} = \frac{\overline{T}}{c'} = 1.39 \log_2 n - 1.85, \quad \text{para } n >> 1 \text{ [3]} \;, \qquad (1.3)$$

onde $\overline{Ac}$ é chamado o número médio de acessos.

O valor máximo de $\overline{Ac}$ é $Ac_{max} = n/2$, que ocorre quando for obtida uma lista na construção da árvore. O valor mínimo é $Ac_{min} = \log_2 n - 1$ [3] para $n >> 1$.

1.4 - Balanceamento

Se a sequência de inserção for aleatória, pode ser obtida uma árvore na qual o número médio de acessos em uma subárvore seja muito maior que na outra. Podem-se, então, adicionar condições aos algoritmos de inserção e eliminação simples para que sejam obtidas sempre árvores balanceadas.

Existem várias técnicas de balanceamento. Cada uma delas realiza um esforço razoável para manter a árvore eficiente após uma inserção ou eliminação. Entre outras técnicas citam-se "AVL-Trees" [1] e Árvores Virtuais [6,7].

"AVL-Trees" é um método de construção e manutenção de árvores binárias em que nenhuma subárvores tem altura (comprimento do maior caminho) com diferença maior que 1 em relação a outra.

Um modelo aproximado para avaliação dos custos de balanceamento com essa técnica é apresentado por C.C.Foster [2].

Árvores Virtuais [7] é uma técnica de construção e manutenção de árvores balanceadas, cujo balanceamento, com uso da distribuição das chaves, é feito da seguinte maneira:

sejam: x - um nó de uma árvore A;
C(x) - a chave real do nó x;
C'(x) - a chave virtual do nó x.

Se as chaves reais forem uniformemente distribuídas no intervalo [1, C_{max}], as chaves virtuais podem ser calculadas por:

$$C'(x) = \frac{2\ pos(x) - 1}{2^{nível(x)+1}} \cdot C_{max}\ , \qquad (1.4)$$

onde pos(x) é a posição do nó x no conjunto de nós de um mesmo nível contando-se da esquerda para a direita, de 1 até $2^{nível(x)}$, ou seja, deve-se levar em conta as posições não usadas;
C_{max} é o valor máximo para as chaves.

A chave virtual não precisa estar armazenada, pois pos(x) e nível (x) são conhecidos.

O balanceamento é obtido quando uma chave real é armazenada em um nó x tal que $|C(x) - C'(x)|$ seja mínimo, considerando todas as chaves virtuais existentes nas subárvores de x.

Exemplo 2 - Considerar a sequência de inserção (200, 350, 550, 800, 900). A árvore obtida está representada na figura 1.2 com C(x)/C'(x) e C_{max} = 1000.

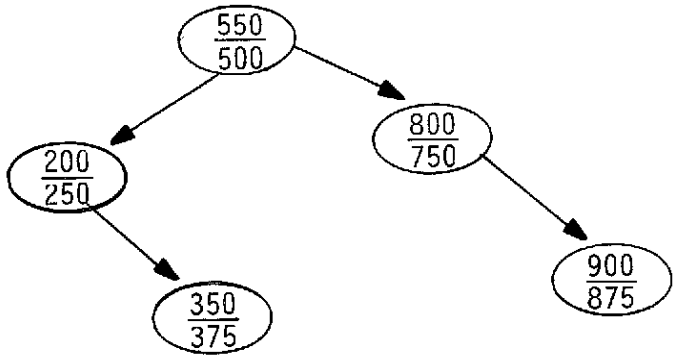

Fig. 1.2 - Árvores balanceadas com a técnica de árvores virtuais.

2 - ÁRVORES BINÁRIAS COM ENDEREÇOS ORDENADOS

Na Seção anterior falou-se de estruturas construídas com base na chave escolhida para raiz e de uma relação binária.

Introduzir-se-á mais um critério para essa construção, o qual definirá uma ordenação nos caminhos de procura [5].

Supõe-se que cada chave C(x) de um nó x de uma árvore binária A possui um enderço E(C(x)). Assim, A, com respeito aos endereços de chaves, é de Endereços Ordenados, se para cada caminho $(x_1, x_2, \ldots, x_k)$ em A for válida, além das condições apresentadas na Equação (1.1), a seguinte relação: $E(C(x_i)) \leq E(C(x_{i+1}))$

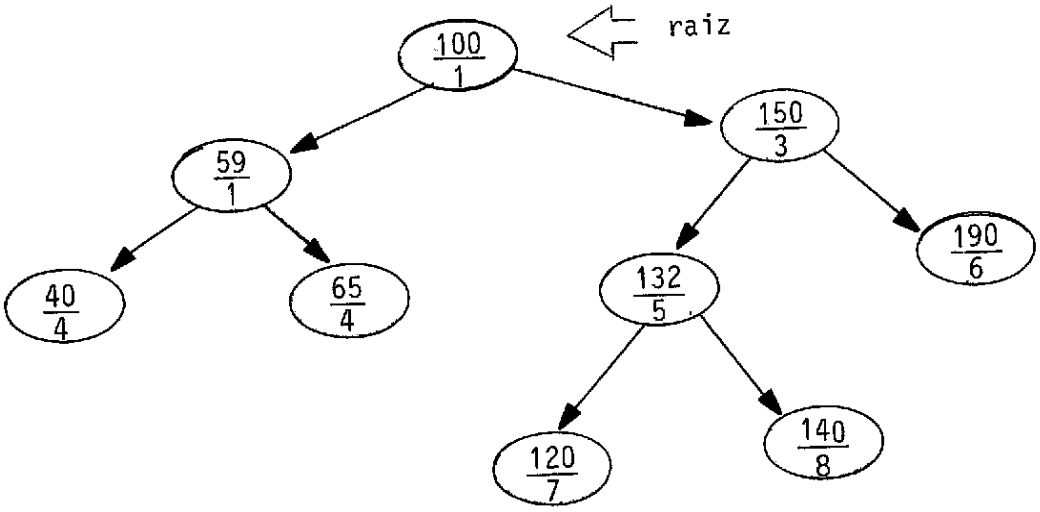

Fig. 2.1 - Árvore Binária com Endereços Ordenados.

2.1 - Algoritmo de Inserção (INS)

Para realizar uma inserção, procede-se de tal maneira que os endereços livres mais próximos da raiz sejam ocupados em primeiro lugar. A nova chave é inserida sempre num nó folha, mas durante o processo de inserção são feitas trocas para que seja mantida a estrutura da árvore e a ordenação dos caminhos. Sejam:

$CP(x) = (y_1, y_2, \ldots, y_k)$ o caminho de procura para o nó x;
E(x) = o endereço de x;
Troca (x, y_i) = troca do nó x por um nó y_i, dada por:
1. $C(x) \leftrightarrow C(y_i)$ - troca de chaves;
2. $Pe(x) \leftrightarrow Pe(y_i)$ - troca dos ponteiros que apontam para as raízes das subárvores esquerdas de x e de y_i;

Esse algoritmo faz com que sejam liberados os nós de endereços maiores (no nível das folhas).

Para que EL seja usado na aplicação da técnica de Árvores Virtuais, deve sofrer as seguintes modificações:

Passo 2: Procura-se MA(x) e ME(x);

Passo 3: Transferir para x a chave de MA(x) ou de ME(x) que estiver mais próxima do valor de C'(x);
Se $|C(MA(x)) - C'(x)| < |C(ME(x)) - C'(x)|$ então C(x): = C(MA(x)) e EL(MA(x)); caso contrário C(x): = C(ME(x)), EL(ME(x)).

Nesse caso, o passo 2 fica mais caro, pois deve processar os dois caminhos para MA(x) e ME(x) e então escolher o nó para substituição.

3 - ÁRVORES BINÁRIAS COM ENDEREÇOS ORDENADOS ARMAZENADAS EM DISCO

Quando grandes árvores binárias estão armazenadas em disco, o principal custo de acesso aos nós é causado pela transferência de dados do disco para a memória principal. Esta transferência é geralmente feita em blocos de tamanho de 100 e 10000 "bytes" [8].

3.1 - Taxa Média de Transferência de Blocos por Árvores Binárias

Seja $(x_1, x_2, \ldots, x_r)$ o caminho de pesquisa para o nó x_r na árvore binária A. O número de blocos a ser transferido é dado por:

$$\overline{b}_A(x_r) = 1 + \sum_{i=1}^{r-1} \theta(x_i, x_{i+1}) , \qquad (3.1)$$

onde

$$\theta(x_i, x_{i+1}) = \begin{cases} 0 \text{ se } x_i \text{ e } x_{i+1} \text{ estiverem no mesmo bloco,} \\ 1 \text{ caso contrário.} \end{cases}$$

Considerando n nós e que todos eles são processados com a mesma frequência, a taxa média de transferência dos blocos da árvore é definida como:

$$\overline{b}_A = \frac{1}{n} \sum_{i=1}^{n} \overline{b}_A(x_i). \qquad (3.2)$$

A taxa média de transferência dos blocos na Equação (3.2) depende ainda da estrutura da árvore A. Considerando que a árvore binária A seja criada através da inserção de n nós aleatórios, a taxa de transferência dos blocos é obtida através da soma sobre todas as árvores A obtidas pelas n! sequência de inserções [3,6], dada por:

$$\bar{b} = \frac{1}{n!} \sum_{i=1}^{n!} \bar{b}_{Ai} \qquad (3.3)$$

Estando os nós distribuídos aleatoriamente sobre os blocos, o custo de um caminho de pesquisa é proporcional ao seu comprimento. Portanto, $\bar{b}$ é proporcional ao número médio de acesso, $\bar{A}_c$, conforme a equação:

$$\bar{b} = \bar{\theta}_a \bar{A}_c ,$$

onde $\bar{\theta}_a$ é o custo médio da taxa de transferência dos blocos correspondentes a um nó e seu sucessor, ou seja, a probabilidade de um nó e seu sucessor em blocos diferentes.

Se n for o número de nós, m o número de nós por bloco e considerando que todos os blocos estejam completamente ocupados por m nós, então:

$$\bar{\theta}_a = 1 - \frac{m-1}{n-1} . \qquad (3.5)$$

3.1 - Taxa Média de Transferência de Blocos para ABEO

Se ABEO forem usadas, é impossível construir-se todas as n! árvores para uma exata determinação da Equação (3.3), portanto $\bar{b}$ foi obtida por simulação:

Passo 1: criar n chaves e endereços aleatórios e construir uma árvore inicial;
Passo 2: executar ℓ inserções e ℓ eliminações aleatórias;
Passo 3: medir $\bar{b}$.

Os passos 2 e 3 devem ser repetidos k $\approx$ 20 vezes a fim de obter um valor aproximado do erro padrão. Os dois parâmetros k e ℓ são escolhidos de modo que o erro padrão seja menor que 5%. O valor de ℓ escolhido foi de n/2 [4].

Os resultados da taxa média de transferência de blocos, $\bar{b}$, para n = 5000 nós em função do tamanho do bloco, são mostrados na Figura 3.1.

Este exemplo demonstra que as ABEO diminuem a taxa média de transferência de blocos pela metade em comparação às árvores comuns. O custo de transferência pouco se altera com o número de nós por bloco. Como pode ser visto, blocos grandes são pouco mais econômicos que blocos pequenos.

O custo para manter uma ABEO com a técnica de Árvores Virtuais é de aproximadamente 2.5 atualizações de blocos no caso de inserção e de 0.6 no caso de eliminação.

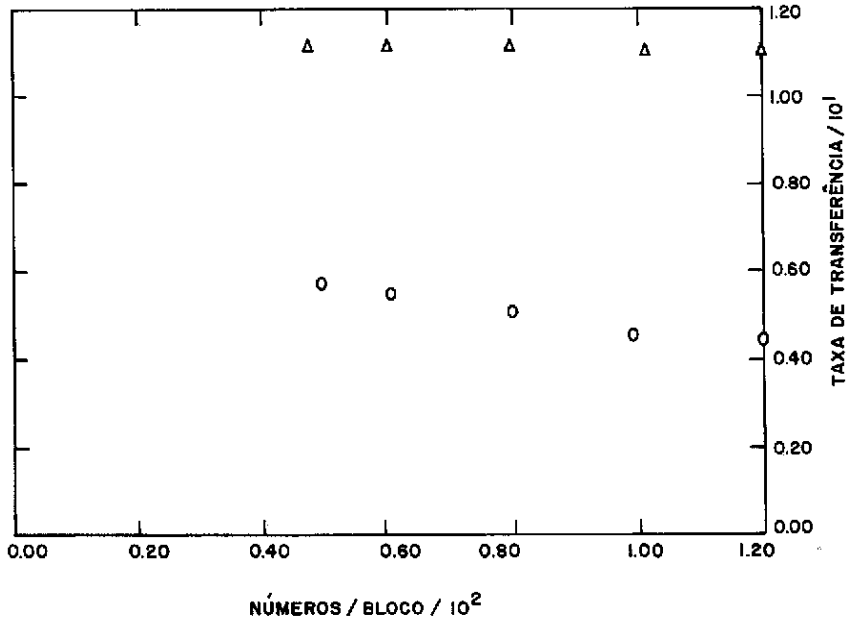

Fig. 3.1 - Taxa média de transferência ($\bar{b}$) em função do número de nós por bloco (m) para árvores comuns (linha contínua) e ABEO (linha trajetória), com número de nós n = 5000.

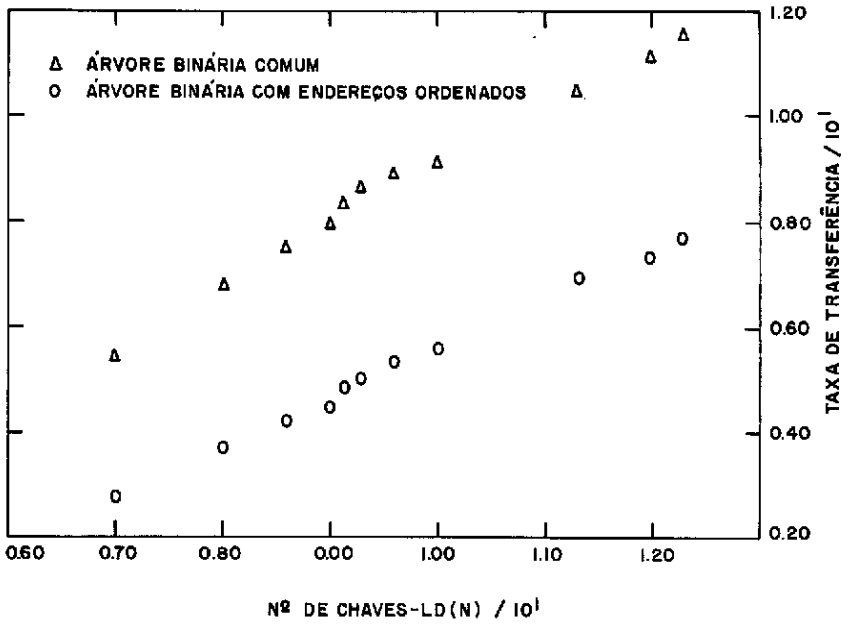

Fig. 3.2 - Taxa média de transferência ($\bar{b}$) em função do número de nós (n) para árvores comuns (+) e ABEO (.), com número de nós por bloco m = 21.

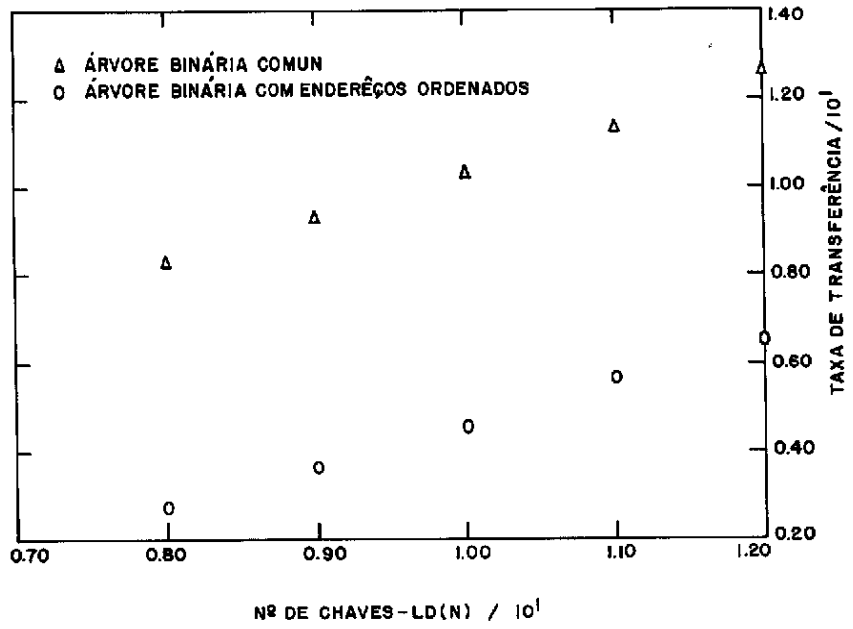

Fig. 3.3 - Taxa média de transferência ($\bar{b}$) na função do número de nós (n) para árvores comuns (+) e ABEO (.), com número de nós por bloco m = 42.

As figuras 3.2 e 3.3 mostram a variação da taxa média de transferência dos blocos em função do número de nós para blocos com 21 e 42 nós respectivamente.

No primeiro caso, o quociente entre o valor de $\bar{b}$ para ABEO e o valor de $\bar{b}$ para árvores comuns é de aproximadamente 60% e no segundo caso é de aproximadamente 50%. Nos dois casos o aumento do quociente é lento quando o número de nós aumenta. O custo de manutenção das ABEO com a técnica de Árvores Virtuais é aproximadamente igual ao mostrado acima.

4 - CONCLUSÃO

Quando grandes árvores binárias estão armazenadas em memória secundária, como disco, a ABEO é uma técnica simples e eficiente para diminuir a taxa de transferência de blocos. O custo de atualização é pequeno e em casos realísticos a taxa de transferência é diminuída pela metade.

O estudo está sendo prolongado para medir o tempo de transferência diretamente nos vários tipos de memória secundária, como por exemplo disco "floppy". Serão computados os resultados com os casos de ABEO sem a técnica de Árvores Virtuais e de ABEO como árvores AVL.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] - ADELSON-VELSKII, G.M. & LANDIS, E.M. "An Information Organization Algoritms", Doklady Akademüa Mauk SSSR, 146:263-266, 1962.

[2] - FOSTER, C.C. "Information Storage and Retrieval Using AVL-Trees", Proceeding of the ACM 20th National Conference, 1(3):173-189, 1972.

[3] - KNUTH, D.E. "The Art of Computer Programming", Massachusetts, Addison Wesley, 1975, v. 1,3.

[4] - PEREIRA, S.A. "Árvores Binárias de Caminhos Ordenados como Modelo de Organização de Arquivos em Memória de Acesso Pseudo-Aleatório". Tese de Mestrado, ITA, 1981.

[5] - SEEHUSEN, J. "Zugriffsgünstige Darstellung von Binarbaumstrukturierten Datein auf Pseudo-Direktzugriffsspeichern". Angenwandte Informatik Berlin, 8:337--342, 1976.

[6] ——— "Virtuelle Binärbaumtechnik". Applied Computer Science, Berlin,1:185:196, 1976.

[7] ——— "Virtuelle Baume und ihre Bedeutung für die Verwaltung gro er Datenbestande". Tese de Doutorado, Berlin, /s.c.p/1976.

[8] - WRIGHT, W.E. "Binary Search Trees in Secondary Memory". Acta Informatica 15, 3-7, 1981.