UM JOGO SOBRE GRAFOS
E
APLICAÇÕES À TEORIA DA COMPLEXIDADE

MIGUEL HADDAD

DISSERTAÇÃO APRESENTADA AO INSTITUTO DE MATEMÁTICA E ESTATÍSTICA DA UNIVERSIDADE DE SÃO PAULO, PARA OBTENÇÃO DO GRAU DE MESTRE EM MATEMÁTICA APLICADA.

ORIENTADOR:

PROF. DR. ISTVAN SIMON

- São Paulo, junho de 1982 -

AGRADECIMENTOS

ao Prof. Dr. Istvan Simon, pela orientação e pelo estímulo constante que tornaram possível a realização deste trabalho e ampliaram em muito meus conhecimentos; a ele e sua esposa Karen, pela maneira carinhosa como ambos me acolheram em minhas recentes viagens a Campinas.

ao Prof. Dr. Cláudio L. Lucchesi, pela colaboração na demonstração do Lema 1.2.3.

ao Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq, pelo auxílio financeiro.

a Regina Helena da Silva e Luzia do Carmo Namiki, pelo excelente trabalho de datilografia e pela extrema boa vontade que demonstraram para comigo;

a Magali Wakabara, Celma de O. Ribeiro, Eliane Dal Bello, Inês S.H. de Melo, Cristina Cerri e, especialmente, a Isabel Helena C. Soares e Raquel de Jesus pela colaboração no trabalho de revisão da datilografia.

a todos os meus amigos, cujo apoio direto ou indireto foi decisivo nos momentos mais difíceis da elaboração deste trabalho.

A minha mãe

e

ao meu irmão

NOTAÇÕES

$f(n) = O(g(n))$	existem constantes $n_o>0$ e $c>0$ tais que para todo $n>n_o$, $f(n) \leq cg(n)$
$f(n) = \Omega(g(n))$	existem constantes $n_o>0$ e $c>0$ tais que para todo $n>n_o$, $f(n) \geq cg(n)$
$f(n) = \Theta(g(n))$	$f(n) = O(g(n))$ e $f(n) = \Omega(g(n))$
$\ell g x$	ℓog_2^x
$\lfloor x \rfloor$	maior inteiro menor ou igual a x
$\lceil x \rceil$	menor inteiro maior ou igual a x
bin(n)	representação binária de n

ÍNDICE

		página
Capítulo 0	- O JOGO DAS PEDRAS	1
Capítulo 1	- GRAFOS E SUPERCONCENTRADORES	
	1. Grafos	6
	2. Superconcentradores	9
Capítulo 2	- O NÚMERO MÍNIMO DE PEDRAS	
	1. Introdução	24
	2. Um Limite Superior para $E_d(n)$	24
	3. Um Limite Inferior para $E_d(n)$	33
Capítulo 3	- TEMPO vs ESPAÇO NO JOGO DAS PEDRAS	
	1. Introdução	42
	2. Um Limite Superior	43
	3. Um Limite Inferior	61
Capítulo 4	- O JOGO DAS PEDRAS E COMPLEXIDADE DE ALGORITMOS	
	1. Introdução	77
	2. O Jogo das Pedras e Máquinas de Turing.	77
	3. Aplicações à Complexidade Concreta	93
Capítulo 5	- CONCLUSÃO	102
REFERÊNCIAS	. . . . .	105

CAPÍTULO 0

O JOGO DAS PEDRAS

Neste trabalho estudaremos o chamado "jogo das pedras", definido em [HP70] e [Wa71].

O jogo é realizado por um único jogador sobre um grafo orientado finito acíclico (vide definição na Seção 1.1). Um lance consiste em *ocupar um vértice*, isto é, colocar uma pedra sobre ele, ou *desocupar um vértice*, isto é, remover uma pedra que está sobre ele. Diremos que um vértice está *ocupado* em dado instante do jogo sse há uma pedra sobre ele.

Os lances estão sujeitos a duas regras:

1. um vértice pode ser desocupado em qualquer instante;
2. um vértice só pode ser ocupado (com uma pedra que não está sobre nenhum vértice) se todos os seus predecessores estão ocupados.

Inicialmente, nenhum vértice está ocupado. O objetivo do jogo é *ocupar o grafo*, isto é, ocupar cada um de seus vértices pe-

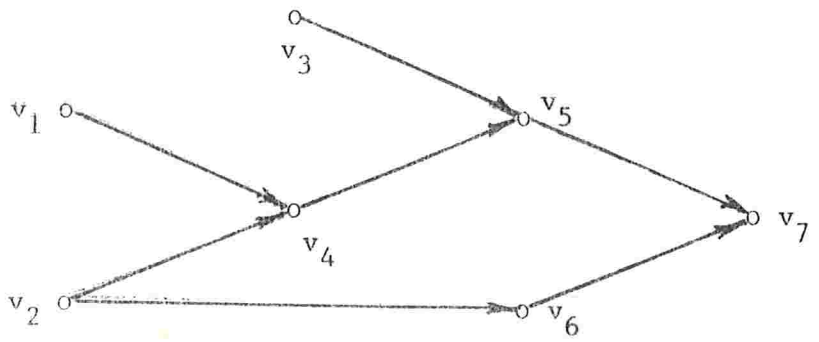


Figura 0.1

lo menos uma vez. Note que não é necessário que todos os vértices do grafo estejam ocupados em algum instante ou ao final do jogo.

Como exemplo vamos apresentar duas estratégias para ocupar o grafo da Figura 0.1, identificando as pedras por 1, 2, 3,...

ESTRATÉGIA 1	ESTRATÉGIA 2
ocupe v_1 com 1	ocupe v_1 com 1
ocupe v_2 com 2	ocupe v_2 com 2
ocupe v_4 com 3	ocupe v_4 com 3
desocupe v_1	desocupe v_1
desocupe v_2	ocupe v_3 com 1
ocupe v_3 com 1	ocupe v_5 com 4
ocupe v_5 com 2	desocupe v_3
desocupe v_3	desocupe v_4
desocupe v_4	ocupe v_6 com 1
ocupe v_2 com 1	ocupe v_7 com 3
ocupe v_6 com 3	
desocupe v_2	
ocupe v_7 com 1	

O número de lances para se ocupar um grafo G usando uma estratégia é o *tempo* da estratégia e número de pedras utilizadas, o *espaço* da estratégia.

No exemplo anterior, a Estratégia 1 gasta tempo 13 e espaço 3 e a Estratégia 2, tempo 10 e espaço 4.

A ocupação de um grafo por uma estratégia pode também ser descrita pela seqüência dos vértices ocupados após cada lance. Um tal cojunto de vértices ocupados é uma *configuração*.

Se for permitido que várias pedras sejam colocadas num

vértice então uma configuração é um multi-conjunto, onde cada vértice tem por multiplicidade o número de pedras que o ocupam.

A *configuração no instante* t da estratégia, C_t, é a configuração imediatamente após o t-ésimo lance e antes do lance seguinte. O *número de pedras no grafo no instante* t é $|C_t|$. Note que as regras do jogo garantem que $C_t = C_{t-1} \cup \{v\}$ ou $C_t \cup \{v\} = C_{t-1}$. No primeiro caso, v é ocupado no instante t e

$$\mathrm{PredG}(\{v\}) \subseteq C_{t-1}$$

(na definição na Seção 1.1). No segundo caso, v é desocupado no instante t.

O jogo foi introduzido por Hewitt & Paterson ([HP70]) no estudo de esquemas recursivos, mas apenas para árvores binárias. Cook ([Co73]) utilizou as mesmas idéias para tentar provar sua conjectura de que $P \subseteq \mathrm{ESPAÇO}(\lg^2)$.

Walker ([Wa71]) generalizou o jogo e aplicou-o ao estudo de cálculo eficiente de expressões. Sethi ([Se73]) utilizou o jogo em alocação eficiente de registradores.

A idéia básica destas duas aplicações e outras posteriores é associar a uma computação um grafo orientado acíclico. Cada vértice representa uma certa informação e as arestas indicam quais informações precisam estar disponíveis para que uma outra seja computada. A ocupação de um vértice representa a computação e o armazenamento da informação correspondente a ele. As fontes do grafo representam as entradas e os sorvedouros, as saídas da computação.

Um problema que surgiu foi o de determinar o espaço $E_d(n)$ necessário para se ocupar qualquer grafo com n vértices e grau de entrada máximo d. Hopcroft, Paul & Valiant ([HPV77]) mostraram que $E_d(n) = O(dn/\ell g\, n)$ e, utilizando este resultado, que para toda função t de N em N, $TEMPO(t) \subseteq ESPAÇO(t/\ell g\, t)$, onde $TEMPO(t)$ e $ESPAÇO(t/\ell g\, t)$ denotam as classes de complexidade de Máquinas de Turing limitadas em tempo por t e em espaço por $t/\ell g\, t$, respectivamente. Paul, Tarjan & Celoni ([PTC77]) mostraram, através da construção de uma família de grafos de difícil ocupação, que $E_2(n) = \Omega(n/\ell g\, n)$ e, portanto,

$$E_d(n) = \theta(n/\ell g\, n).$$

Um problema que despertou interesse, então, foi o de determinar o tempo T necessário para se ocupar um grafo com n vértices e grau de entrada máximo d se fixarmos o espaço E disponível. Várias tentativas foram feitas para resolver este problema ([PT77], [EL78], [Li78], [Rei78]) até que Lengauer ([Le79]) mostrou que

$$T = E2^{2^{\theta(n/E)}}.$$

Mais recentemente, o jogo tem sido utilizado para determinar relações entre tempo e espaço em Complexidade Concreta ([Va76], [To78], [Ja80]).

O Capítulo 1 deste trabalho é dedicado à Teoria dos Grafos. Na Seção 1.1 introduzimos alguns conceitos básicos sobre grafos orientados, visando principalmente apresentar a nomenclatura e a notação que adotaremos. Na Seção 1.2 es-

tudamos uma família de grafos chamados superconcentradores que são de grande utilidade na determinação de limites inferiores em alguns problemas relacionados com o Jogo das Pedras.

No Capítulo 2 estudamos as funções $E_d(n)$ e no Capítulo 3, o trabalho de Lengauer sobre tempo vs. espaço no jogo.

Finalmente, no Capítulo 4 apresentamos algumas aplicações do jogo à Teoria da Complexidade.

- ▯ -

CAPÍTULO 1

GRAFOS E SUPERCONCENTRADORES

1 - GRAFOS

Um *grafo orientado* G consiste de um conjunto VG de elementos chamados *vértices*, um conjunto aG de elementos chamado *arestas* e um par ordenado $(\iota G, \phi G)$ de funções de aG em VG.

Para todo $\alpha \in aG$, $\iota G(\alpha)$ é chamado *extremo inicial* de α e $\phi G(\alpha)$, *extremo final* de α. Dizemos ainda que α *sai* de $\iota G(\alpha)$ e *chega* em $\phi G(\alpha)$.

Um grafo orientado G é *finito* se VG e aG são finitos. Pelo termo "grafo" entenderemos um grafo orientado finito.

Sobre VG vamos definir uma relação que chamaremos de *relação de precedência:*

$$u \longrightarrow_G v \text{ sse existe } \alpha \text{ em } aG \text{ tal que}$$

$$\iota G(\alpha) = u \quad \text{e} \quad \phi G(\alpha) = v.$$

Se $u \longrightarrow_G v$ dizemos que u é *predecessor* de v e v é *sucessor* de u. Se $\iota G(\alpha) = \phi G(\alpha)$ dizemos que α é um laço. Caso contrário, α é uma *ligação*.

O *tamanho* de um grafo G é o inteiro $|VG| + |aG|$.

Para todo $V' \subseteq VG$, denotaremos por SucG(V') o conjunto dos

sucessores dos vértices de V', isto é,

$$SucG(V') = \left\{v \in VG: \exists u \in V' \mid u \rightarrow_G v\right\}.$$

Analogamente,

$$PredG(V') = \left\{v \in VG: \exists u \in V' \mid v \rightarrow_G u\right\}.$$

Para todo $V', V'' \subseteq VG$, denotaremos por $aG(V', V'')$ o conjunto das arestas que saem de algum vértice de V' e chegam em algum vértice de V'', isto é,

$$aG(V', V'') = \left\{\alpha \in aG: \iota G(\alpha) \in V' \text{ e } \phi G(\alpha) \in V''\right\}.$$

Um grafo G é *bipartido* se existe uma partição $\{V', V''\}$ de VG tal que $aG(V', V'') = aG$.

Um grafo G' é *subgrafo* de um grafo G, $G' \subseteq G$, se $VG' \subseteq VG$, $aG' \subseteq aG$ e $\iota G'$ e $\phi G'$ são, respectivamente, as restrições de ιG e ϕG a aG'.

Para todo $V' \subseteq VG$, o *subgrafo* de G *gerado* por V', $G[V']$, é o subgrafo G' de G tal que $VG' = V'$ e $aG' = aG(V', V')$.

O *grau de entrada* (*grau de saída*) de um vértice v de G é o número de arestas que chegam em (saem de) v. Uma *fonte* (*sorvedouro*) é um vértice com grau de entrada (saída) nulo.

Um grafo G é *simples* se não possui laços nem arestas múltiplas, isto é, arestas distintas com mesmo extremo inicial e mesmo extremo final.

Um *passeio* em G é uma seqüência finita não vazia $(v_0, \alpha_1, v_1, \alpha_2, v_2, \ldots, \alpha_n, v_n)$ $(n \geq 0)$ tal que $v_0, v_1, \ldots, v_n \in VG$, $\alpha_1, \ldots, \alpha_n \in aG$, e para todo j $(1 \leq j \leq n)$ $\iota G(\alpha_j) = v_{j-1}$ e $\phi G(\alpha_j) = v_j$ ou $\iota G(\alpha_j) = v_j$ e $\phi G(\alpha_j) = v_{j-1}$. Dizemos que este passeio *liga* v_0 a v_n. Se

$$v_0 \in V' \subseteq VG \text{ e } v_n \in V'' \subseteq VG,$$

dizemos que o passeio liga V' a V". O inteiro n é chamado *comprimento do passeio*.

Um grafo é *conexo* se quaisquer dois de seus vértices são ligados por um passeio. Se um grafo não é conexo, cada um de seus subgrafos conexos maximais é chamado *componente* do grafo.

Um *caminho orientado* em G é um passeio $(v_0,\alpha_1,v_1,\ldots,\alpha_n,v_n)$ tal que $v_0,v_1,\ldots,v_n$ são dois a dois distintos e para todo j $(1\leq j\leq n)$ $\iota G(\alpha_j) = v_{j-1}$ e $\phi G(\alpha_j) = v_j$. Por simplificação, vamos nos referir a um caminho orientado apenas por "caminho".

Dois caminhos são *disjuntos* se não tem um vértice em comum. Um *k-fluxo* é um conjunto de k caminhos dois a dois disjuntos.

Um *circuito* em G é um passeio $(v_0,\alpha_1,v_1,\ldots,\alpha_n,v_n)$ de comprimento $n \geq 1$ tal que $(v_0,\alpha_1,v_1,\ldots,\alpha_{n-1},v_{n-1})$ é um caminho, $\iota G(\alpha_n) = v_{n-1}$, $\phi G(\alpha_n) = v_n$ e $v_0 = v_n$. Um grafo é *acíclico* se não possui circuitos Por G(n,d) denotaremos a família dos grafos acíclicos com n vértices e grau de entrada máximo menor ou igual a d. A não ser que esteja explícito o contrário, suporemos neste trabalho que $d \geq 2$.

Uma ordenação $v_1,v_2,\ldots,v_{|VG|}$ dos vértices de G é *topológica* se para todo i, j $(1\leq i,j\leq |VG|)$ $v_i \rightarrow_G v_j$ implica $i > j$.

Dois resultados básicos sobre grafos acíclicos serão muito utilizados por nós.

TEOREMA 1 - Qualquer grafo (finito) acíclico tem pelo menos uma fonte e pelo menos um sorvedouro. □

TEOREMA 2 - Um grafo (finito) G é acíclico sse G admite uma or-

denação topológica de seus vértices. □

As demonstrações serão deixadas a cargo do leitor.

2 - SUPERCONCENTRADORES

Uma família de grafos que tem especial interesse no estudo do Jogo das Pedras é a dos chamados superconcentradores.

Um (N,M)-*superconcentrador* é um grafo acíclico com N fontes e M sorvedouros tal que para todo k ($1 \leq k \leq N,M$), quaisquer k fontes do grafo estão ligadas a quaisquer k sorvedouros por um k-fluxo.

Quando N = M chamaremos o grafo abreviadamente de N-*superconcentrador*.

Superconcentradores tem sido muito utilizados na determinação de limites inferiores no Jogo das Pedras e em algumas de suas aplicações. A utilização se baseia, principalmente, nos resultados que apresentaremos mais adiante.

Um caminho num grafo G está *livre de pedras* num instante t se nenhum de seus vértices está em C_t.

Por "os vértices de $V' \subseteq VG$ são ocupados durante um intervalo T" entendemos que para todo $v \in V'$, existe $t \in T$ tal que $v \in C_t$, mas não necessariamente existe $t' \in T$ tal que $V' \subseteq C_{t'}$.

LEMA 1 - Para todo E ($0 \leq E < M,N$), se E vértices de um (N,M)-superconcentrados S estão ocupados num instante t e Y é um conjunto de E+1 sorvedouros de S, então existem N-E fontes de S ligadas a Y por algum caminho livre de pedras no instante t.

DEMONSTRAÇÃO - Qualquer conjunto X de E+1 fontes de S é liga-

do a Y por um (E+1)-fluxo e, portanto, pelo menos uma fonte de X está ligada a Y por um caminho livre de pedras no instante t.

Logo, o conjunto das fontes de S que não estão ligadas a Y por um caminho livre de pedras no instante t tem no máximo E elementos.

LEMA 2 (Lema Fundamental) - Para todo E ($0 \leq E < N,M$), se E+1 sorvedouros de um (N,M)-superconcentrador S são ocupados durante um intervalo $T = [t_1, t_2]$ tal que $|C_{t_1-1}| = E$ então pelo menos N-E fontes de S (que não estão em C_{t_1-1}) são ocupadas durante T.

DEMONSTRAÇÃO - Pelo Lema 1, no instante t_1-1 existem pelo menos N-E fontes de S ligadas aos E+1 sorvedouros por algum caminho livre de pedras. Durante T cada uma destas fontes é ocupada pelo menos uma vez. □

O Lema Fundamental pode se apresentar sob outra forma.

LEMA 3 - Se S é um (N,M)-superconcentrador então são equivalentes:

1. O Lema Fundamental.
2. Para todo E_i, E_f ($0 \leq E_i, E_f$, $0 \leq E_i + E_f < N,M$), se E_i+E_f+1 sorvedouros de S são ocupados durante um intervalo $T = [t_1, t_2]$ tal que $|C_{t_1-1}| = E_i$ e $|C_{t_2}| = E_f$, então pelo menos $N-E_i-E_f$ fontes de S são ocupadas e desocupadas durante T.

DEMONSTRAÇÃO - 1 $\Longrightarrow$ 2 - Sejam E_i, E_f e T nas condições de 2.

Seja $T' \subseteq T = [t_1, t_2]$ o sub-intervalo de T que compreende os instantes de t_1 até aquele em que o (E_i+1)-ésimo sorvedouro de

S é ocupado. Aplicando 1. a T' temos que durante $T' \subseteq T$ pelo menos $N-E_i$ fontes de S são ocupadas. Como apenas E_f vértices de S estão ocupados ao final de T, pelo menos $N-E_i-E_j$ daquelas fontes são desocupadas durante T.

$2 \Longrightarrow 1$ - Sejam E, T nas condições de 1.

Seja j o número de pedras em S ao final de T, isto é, $j = |C_{t_2}|$.

Removendo estas pedras do grafo, obtemos um intervalo $[t_1, t_2+j]$ durante o qual E+1 sorvedouros de S são ocupados, $|C_{t_1-1}| = E$ e $|C_{t_2+j}| = 0$.

Por 2., N-E fontes de S são ocupadas e desocupadas durante $[t_1, t_2+j]$. Como só há desocupações durante $[t_2+1, t_2+j]$, as N-E fontes são ocupadas durante $[t_1, t_2]$. □

LEMA 4 - Se todos os sorvedouros de um (N,M)-superconcentrador S são ocupados durante um intervalo [1,T] em espaço E então

$$T \geq E + M(N-E)/(E+1).$$

DEMONSTRAÇÃO - Vamos supor, sem perda de generalidade, que cada sorvedouro é ocupado apenas uma vez.

Para $i = 1, 2, \ldots, M$, seja t_i o instante em que o i-ésimo sorvedouro é ocupado.

Como S é um (N,M)-superconcentrador, cada fonte de S é ligada a cada um dos seus sorvendouros. Assim, durante $[1, t_1-1]$ todas as fontes de S são ocupadas e, portanto, $t_1 - 1 \geq N$.

Durante $[t_1+1, t_{E+2}]$, E+1 sorvedouros de S são ocupados. Como $|C_{t_1}| \leq E$, segue do Lema Fundamental que durante $[t_1+1, t_{E+2}]$ pelo menos N-E fontes de C são ocupadas e, portanto,

$$t_{E+2}-t_1 \geq N-E.$$

Aplicando este raciocínio sucessivamente temos:

$$\begin{aligned} T &\geq N + (N-E)\lfloor (M-1)/(E+1) \rfloor \\ &\geq N + (N-E)(M-E-1)/(E+1) \\ &= E + (N-E)[1+(M-E-1)/(E+1)] \\ &= E + M(M-E)/(E+1). \end{aligned}$$

□

Uma família $\{S_N\}_{N\geq 1}$ de N-superconcentradores é *linear* se:

1. existe um natural d tal que para todo N ($N\geq 1$), o grau de entrada máximo de S_N é menor ou igual a d;
2. existe uma constante positiva c tal que para todo $N(N\geq 1)$ $|VS_N| \leq cN$.

Valiant ([Va76]) mostrou que existem famílias lineares de N-superconcentradores, mas sua demonstração não é construtiva. Posteriormente, Pippenger ([Pi77]) construiu uma família linear de N-superconcentradores, mas um dos passos da construção baseia-se em argumentos de contagem. Esta construção será vista mais adiante. Gabber & Galil ([GG79]) construiram explicitamente N-superconcentradores, para N quadrado perfeito.

Sejam N, d, c_1, c_2 inteiros positivos tais que $\frac{1}{2} \leq c_1/c_2 < 1$, $c_1 \geq 2$ e c_1 divide $c_2 d$. Seja $d' = c_2 d/c_1$. Um (N,d,c_1,c_2)-*concentrador* é um grafo bipartido com N fontes e $c_1 \lceil N/c_2 \rceil$ sorvedouros tal que:

1. cada fonte tem grau de saída no máximo d;
2. cada sorvedouro tem grau de entrada no máximo d';

3. todo conjunto X de fontes do grafo tal que $|X| \leq N/2$ está ligado a pelo menos $|X|$ sorvedouros.

LEMA 5 - Se X é um conjunto de fontes de um (N,d,c_1,c_2)-concentrador C tal que $|X| \leq N/2$ então existe um $|X|$-fluxo de X para algum conjunto de sorvedouros de C.

DEMONSTRAÇÃO - Seja $Y = SucC(X)$. Da definição de concentrador segue que $|X'| \leq |SucC(X')|$, para todo $X' \subseteq X$.

Como $SucC(X') \subseteq Suc(X) = Y$, para todo $X' \subseteq X$, aplicando o Teorema de Hall ao subgrafo $C[X \cup Y]$ de C temos que existe um emparelhamento neste grafo que cobre X e, portanto, um $|X|$-fluxo de X para algum subconjunto de Y. □

Concentradores podem ser usados para uma construção recursiva de famílias lineares de N-superconcentradores.

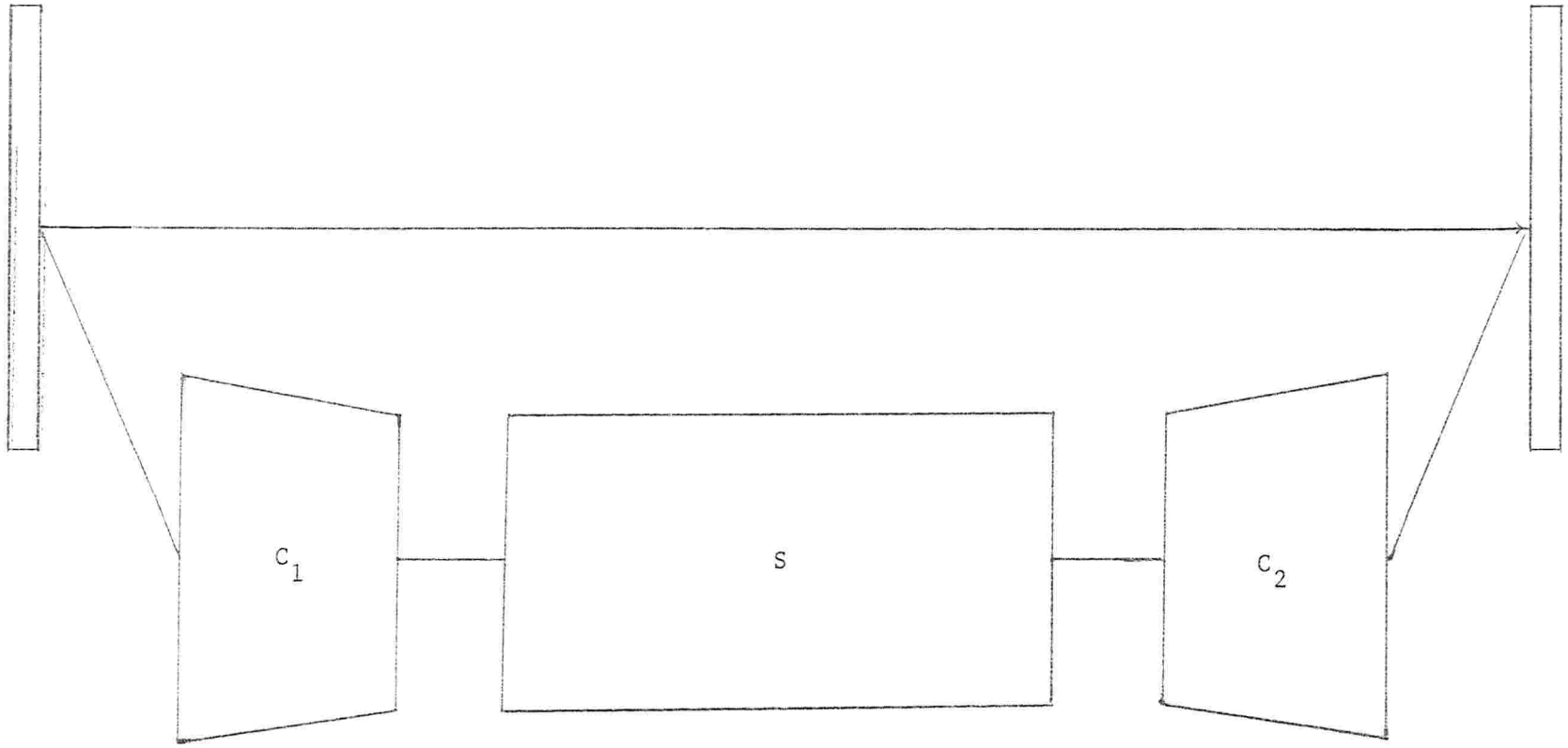
Seja $\lambda(N) = c_1 \lceil N/c_2 \rceil$. Seja $k = c_1(c_2-1)/(c_2-c_1)$. É fácil mostrar que $k \geq 3$ que para todo $N > k$, $\lambda(N) < N$.

Vamos construir recursivamente uma família $\{S_N\}_{N \geq 1}$ da seguinte maneira:

1. se $N \leq k$, S_N é o grafo bipartido simples com N fontes e N sorvedouros tal que toda fonte é ligada a todos os sorvedouros;
2. se $N > k$, S_N é composto por uma cópia C_1 de um (N,d,c_1,c_2)-concentrador, uma cópia C_2 de C_1 com as as arestas invertidas e uma cópia S de um $\lambda(N)$-superconcentrador, ligadas como na Figura 1.1.

TEOREMA 6 - Para todo N $(N \geq 1)$, S_N é um N-superconcentrador.

DEMONSTRAÇÃO - Por indução sobre N.



⟶ indica arestas ligando vértices correspondentes

—— indica identificação de vértices

Figura 1.1

Para $N \leq k$, o teorema é evidentemente válido.

Para $N > k$, seja X um conjunto de fontes de S_N e seja Y um conjunto de sorvedouros de Y tais que $|X| = |Y|$. Vamos mostrar que existe um $|X|$-fluxo de X para Y.

Seja Y' o conjunto dos sorvedouros de S_N correspondentes às fontes de X e seja X' o conjunto das fontes de X cujo sorvedouro correspondente está em $Y \cap Y'$. É imediato que $|X'| = |Y \cap Y'|$ e $|X-X'| = |Y-Y'| \leq N/2$. Também é imediato que existe um $|X'|$-fluxo de X' para $Y \cap Y'$ (composto por caminhos de comprimento 1). Para obtermos o teorema, basta mostrar que existe um $|X-X'|$-fluxo de $X-X'$ para $Y-Y'$.

Pelo Lema 5, existe um $|X-X'|$-fluxo de $X-X'$ para um conjunto X'' de $|X-X'|$ fontes de S e outro de um conjunto Y'' de sorvedouros de S para $Y-Y'$. Como S é um superconcentrador, existe um $|X-X'|$-fluxo de X'' para Y'' em S. A concatenação destes fluxos nos dá um $|X-X'|$-fluxo de $X-X'$ para $Y-Y'$. □

Para i inteiro positivo, vamos definir $\lambda^i(N)$ por

$$\lambda^1(N) = \lambda(N) \text{ e } \lambda^i(N) = \lambda(\lambda^{i-1}(N)), \text{ para } i > 1.$$

LEMA 7 - Para todo $i \geq 1$ e todo $N \geq 1$,

$$\lambda^i(N) \leq (c_1/c_2)^i N + \frac{c_1}{c_2}(c_2-1)\,\frac{1-(c_1/c_2)^i}{1-(c_1/c_2)}$$

DEMONSTRAÇÃO - Por indução sobre i. □

PROPOSIÇÃO 8 - Para todo $i \geq 1$ e todo $N \geq 1$,

$$\lambda^i(N) < (c_1/c_2)^i N + k$$

DEMONSTRAÇÃO - Segue do Lema 7. □

TEOREMA 9 - Para todo $N \geq 1$,

$$|VS_N| \leq \left(\frac{1}{1-c_1/c_2} + 1 + \frac{\ell g\ k}{\ell g(c_2/c_1)}\right) 2N$$

DEMONSTRAÇÃO - Por indução sobre N.

Para $N \leq k$, $|VS_N| = 2N$ e, portanto, a tese é válida.

Para $N > k$, temos $|VS_N| = 2N + |VS_{\lambda(N)}|$. Aplicando repetidamente esta fórmula obtemos:

$$|VS_N| = 2(N + \lambda(N) + \lambda^2(N) + \ldots + \lambda^r(N)),$$

onde r é o menor inteiro tal que

$$\lambda^r(N) \leq k. \tag{1}$$

Pela Proposição 8,

$$\begin{aligned} |VS_N| &< 2N \sum_{i=0}^{r-1} (c_1/c_2)^i + 2rk \\ &= 2N \frac{1-(c_1/c_2)^r}{1-c_1/c_2} + 2rk \\ &\leq 2N \frac{1}{1-c_1/c_2} + 2rk. \end{aligned}$$

Tomando

$$r_0 = \left\lceil \frac{\ell g N}{\ell g(c_2/c_1)} \right\rceil$$

temos $(c_1/c_2)^{r_0} N < 1$. Pela Proposição 8, $\lambda^{r_0}(N) < k+1$, isto é, $\lambda^{r_0}(N) \leq k$. Logo,

$$r \le r_0 = \left\lceil \frac{\ell g\, N}{\ell g(c_2/c_1)} \right\rceil \le 1 + \frac{\ell g\, N}{\ell g(c_2/c_1)}.$$

Como $N > k \ge 3$, $N/\ell g\, N$ é crescente e, portanto, $k\,\ell g\, N \le N\,\ell g\, k$. Assim,

$$kr \le k\left(1 + \frac{\ell g\, N}{\ell g(c_2/c_1)}\right) \le k + \frac{k\,\ell g\, N}{\ell g(c_2/c_1)} \le N + \frac{N\,\ell g\, k}{\ell g(c_2/c_1)}$$

$$= \left(1 + \frac{\ell g\, k}{\ell g(c_2/c_1)}\right) N.$$

Portanto,

$$|VS_N| \le \left(\frac{1}{1-c_1/c_2} + 1 + \frac{\ell g\, k}{\ell g(c_2/c_1)}\right) 2N. \qquad \square$$

COROLÁRIO 10 - A família $\{S_N\}_{N\ge 1}$ é uma família linear de N-superconcentradores.

DEMONSTRAÇÃO - Segue do Teorema 9 e de que todo S_N tem grau de entrada máximo limitado superiormente por $\max\{d',k\}$. $\square$

Veremos a seguir o resultado de Pippenger sobre a existência de $(N,6,4,6)$-concentradores.

TEOREMA 11 - Para todo $j \ge 1$, existe um $(6j,6,4,6)$-concentrador.

DEMONSTRAÇÃO - Para cada permutação π de $D = \{0,1,\ldots,36j-1\}$, vamos definir o grafo $G(\pi)$ como sendo o grafo simples bipartido com $6j$ fontes $\{(1,z): 0 \le z \le 6j-1\}$ e $4j$ sorvedouros $\{(2,z): 0 \le z \le 4j-1\}$, tal que $(1, z \bmod 6j) \rightarrow_{G(\pi)} (2, \pi(z) \bmod 4j)$, para todo $z \in D$. Cada fonte de $G(\pi)$ tem grau de saída no máximo 6 e cada sorve-

douro grau de entrada no máximo 9.

Diremos que $G(\Pi)$ é *bom* se não existem um conjunto X de i fontes ($i \leq 3j$) e um conjunto Y de i sorvedouros de $G(\Pi)$ tais que toda aresta que sai de X chega em Y. Note que se $G(\Pi)$ é bom, então $G(\Pi)$ é um $(6j,6,4,6)$ - concentrador.

Mostraremos que existe Π tal que $G(\Pi)$ é bom determinando um limite superior menor que 1 para a razão entre o número de permutações cujo grafo não é bom e o número total de permutações.

Cada conjunto X de i fontes corresponde a um conjunto A de 6i elementos de D e cada conjunto Y de i sorvedouros corresponde a um conjunto B de 9i elementos de D. Se todas as arestas que saem de X chegam em Y é porque Π leva cada elemento de A num elemento de B.

Para cada A e B, das $(36j)!$ permutações de D existem $(9i)(9i-1)\ldots(9i-6i+1)(36j-6i)!$ que satisfazem a condição acima. Para cada i, existem $\binom{6j}{i}$ escolhas para A e $\binom{4j}{i}$ escolhas para B. Assim, a razão entre o número de permutações cujo grafo não é bom e o número total de permutações é, no máximo,

$$I_j = \sum_{i=1}^{3j} \binom{6j}{i}\binom{4j}{i} \frac{(9i)(9i-1)\ldots(9i-6i+1)(36j-6i)!}{(36j)!} =$$

$$= \sum_{i=1}^{3j} \frac{\binom{6j}{i}\binom{4j}{i}\binom{9i}{6i}}{\binom{36j}{6i}}$$

O número de maneiras de escolher 6i entre 36j objetos é maior ou igual ao número de maneiras de escolher i objetos en

tre os primeiros 6j, i entre os 4j seguintes e 4i entre os últimos 26j. Assim,

$$\binom{36j}{6i} \geq \binom{6j}{i}\binom{4j}{i}\binom{26j}{4i}$$

Daí segue que

$$I_j \leq J_j = \sum_{i=1}^{3j} \binom{9i}{6i}/\binom{26j}{4i}$$

Vamos determinar o maior termo de J_j. Para cada i, seja $L_i = \binom{9i}{6i}/\binom{26j}{4i}$. Escrevendo a razão L_{i+1}/L_i na forma

$$\frac{L_{i+1}}{L_i} = \frac{(9i+9)\ldots(9i+7)\,(9i+6)\ldots(9i+1)\,(4i+4)\,(4i+3)\ldots(4i+1)}{(6i+6)\ldots(6i+1)\qquad(3i+3)\ldots(3i+1)\,(26j-4i)\ldots(26j-4i-3)}$$

podemos verificar que ela cresce com i e, portanto, o maior termo de J_j é L_1 ou L_{3_j}

Se o maior termo for L_1 então

$$J_j \leq 3_j\, L_1 = \frac{3j\,\binom{9}{6}}{\binom{26j}{4}} = \frac{3024}{13(26j-1)(26j-2)(26j-3)} \leq$$

$$\leq \frac{3024}{13\times 25\times 24\times 23} < 1$$

Se o maior termo for L_{3_j} então

$$J_j \leq \frac{3j\,\binom{27j}{18j}}{\binom{26j}{12j}} = 3j\,\frac{(27j)!(12j)!(14j)!}{(18j)!(9j)!(26j)!}$$

Usando a fórmula de Stirling na forma

$$(2\Pi n)^{1/2}\ e^{-n}\ n^n \leq n! \leq e^{1/12n}\ (2\Pi n)^{1/2}\ e^{-n}\ n^n$$

e

$$e^z < \frac{1}{1-z}, \qquad \text{para } z<1$$

obtemos

$$n! \leq \frac{12n}{12n-1}\ (2\Pi n)^{1/2}\ e^{-n}\ n^n$$

Substituindo na desigualdade acima temos

$$J_j \leq 3j\ L_{3j}$$

$$\leq 3j\ \frac{324j}{324j-1}\ \frac{144j}{144j-1}\ \frac{168j}{168j-1}\left(\frac{27\times12\times14}{26\times\ 9\times18}\right)^{1/2}$$

$$\left(\frac{27^{27}12^{12}14^{14}}{26^{26}18^{18}\ 9^{9}}\right)^{j}$$

Podemos verificar que $J_1<1$ e $J_2<1$ calculando-os. Para $j=3$, podemos verificar que o limite superior acima é menor que 1 calculando-o diretamente. Para $j>3$, o limite também é menor que 1 já que se j aumenta de 1, o primeiro fator aumenta de um fator de no máximo 4/3, os três fatores seguintes decrescem e o último fator decresce de um fator maior que 2.

Assim, para todo $j\geq1$, $I_j<1$ e, portanto, existe uma permutação Π de D tal que $G(\Pi)$ é bom, i.e., é um $(6j,6,4,6)$-concentrador. □

COROLÁRIO 12 - Para todo N ($N \geq 1$), existe um (N,6,4,6)-concentrador.

DEMONSTRAÇÃO - Se $N \leq 6$, basta tomar o grafo bipartido com N fontes e 4 sorvedouros tal que toda fonte é ligada a todos os sorvedouros.

Se $N > 6$, seja $m \geq 2$ tal que $6(m-1) < N \leq 6m$ e seja $j = 6m - N$. Seja C um (6m,6,4,6)-concentrador. A remoção de j fontes de C gera um (N,6,4,6)-concentrador, a não ser que um de seus sorvedouros se torne uma fonte (se todos os seus predecessores estão entre as fontes removidas). No entanto, veremos que é possível contornar este problema. Vamos considerar dois casos.

Caso 1: existem j sorvedouros em C com grau de entrada maior ou igual a j+1.

Para obter o (N,6,4,6)-concentrador, basta remover j predecessores desses sorvedouros.

Caso 2: o número de sorvedouros de C com grau de entrada maior ou igual a j+1 é no máximo j-1. □

O número de arestas de C é no máximo $j(4m-j+1)+9(j-1)$. Como j é um inteiro entre 0 e 5, podemos verificar facilmente que este valor é máximo para $j=5$. Como $m \geq 2$, poderíamos acrescentar a C até $36m-|5(4m-4)+36| = 16m-16 \geq 4m$ arestas, obtendo ainda um (6m,6,4,6)-concentrador. Acrescentando um número suficientemente grande de arestas recaímos no Caso 1. □

Os resultados de Pippenger nos garantem a existência de uma família linear de N-superconcentradores $\{S_N\}_{N>1}$ com grau de entrada máximo 10 e no máximo 22N vértices (vide Teorema 10)

Para o estudo de limites inferiores seria interessante que

tivéssemos N-superconcentradores com qualquer grau de entrada máximo . Em particular, N-superconcentradores com grau de entrada máximo 2 já que $G(n,2) \subseteq G(n,d)$, para todo $n \geq 1$ e todo $d \geq 2$.

É possível construir tais N-superconcentradores a partir dos de Pippenger, como no teorema abaixo.

TEOREMA 13 - Para todo $N \geq 1$, existe um N-superconcentrador $\bar{S}_N \in G(190N,2)$.

DEMONSTRAÇÃO - Seja $N \geq 1$ e consideremos o N-superconcentrador S_N de Pippenger.

Vamos supor que S_N tenha um vértice v com grau de entrada $d'>2$. Substituindo as arestas que chegam em v por uma árvore binária com d' folhas e identificando v com a raiz da árvore e seus predecessores com as folhas, obtemos um N-superconcentrador com $d'-2$ vértices a mais que S_N e no qual v tem grau de entrada 2. A Figura 1.2 ilustra esta operação para $d'=5$.

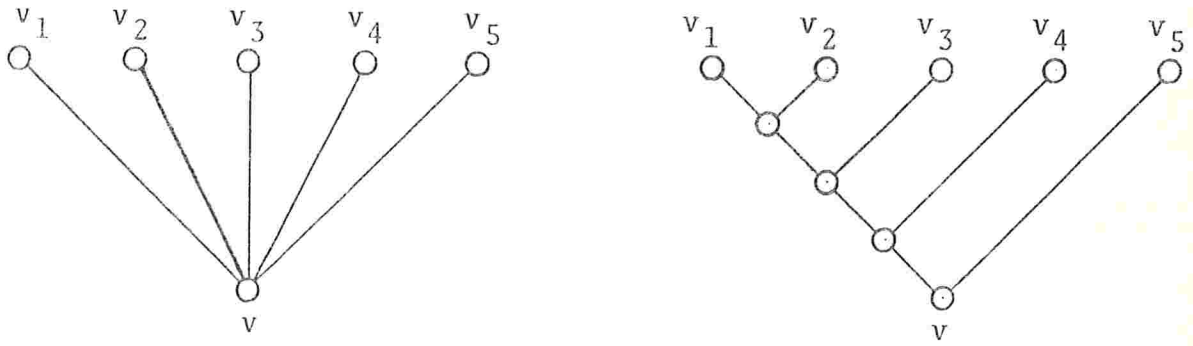


Figura 1.2

Note que cada vértice acrescentado tem grau de entrada 2.

Repetindo esta operação para todo vértice de S_N com grau de entrada (em S_N) maior que 2 obtemos um N-superconcentrador com grau de entrada máximo 2. Este grafo tem, no máximo, $22N + 8(22N-N) = 190N$ vértices. Substituindo um caminho de comprimento 1 por um caminho suficientemente comprido obtemos $\bar{S}_N$.

□

CAPÍTULO 2

O NÚMERO MÍNIMO DE PEDRAS

1 - INTRODUÇÃO

Muitas das aplicações do Jogo das Pedras sugerem o problema de determinar o espaço $E_d(n)$ necessário para se ocupar qualquer grafo de $G(n,d)$, para $d \geq 2$. Para $d = 1$, é imediato que $E_d(n) = 2$, para todo $n \geq 1$.

Cook ([Co73])mostrou que $E_2(n) = \Omega(n^{1/2})$. Posteriormente Hopcroft, Paul & Valiant ([HPV77]) mostraram que

$$E_d(n) = O(dn/\ell g\ n).$$

De sua demonstração podemos derivar um algoritmo que ocupa qualquer grafo de $G(n,d)$ em espaço $O(dn/\ell g\ n)$, que é apresentado na Seção 2 deste capítulo.

Paul, Tarjan & Celoni ([PTC77]) construiram, para todo $n \geq 1$, um grafo de $G(n,2)$ cuja ocupação requer espaço $\Omega(n/\ell g\ n)$. Assim, o algoritmo apresentado na Seção 2 é ótimo a menos de um fator constante, isto é, $E_d(n) = \theta(n/\ell g\ n)$.

2 - UM LIMITE SUPERIOR PARA $E_d(n)$

Nesta seção vamos apresentar um algoritmo que chamaremos MELHOR-OCUPAÇÃO e que ocupa qualquer grafo G de $G(n,d)$ com $O(dn/\ell g\ n)$ pedras.

Inicialmente, vamos descrevê-lo informalmente.

Se G é "pequeno", seus vértices são ocupados em alguma ordem topológica. Após a ocupação de um vértice v qualquer, todos os vértices ocupados cujos sucessores em G são anteriores a v ou o próprio v são desocupados.

Se G é "grande", particionamos VG em dois subconjuntos VG' e VG" tais que

$$\left.\begin{array}{l} 1.\quad aG(VG'',VG') = \emptyset; \\ 2.\quad m/2 + m/\ell gm \leq |aG(VG,VG'')| \leq m/2 + m/\ell gm+d, \\ \quad\ \text{onde } m = |aG| \end{array}\right\} \quad (1)$$

Como {VG',VG"} é uma partição de VG, segue que $VG' \cup VG'' = VG$ e $VG' \cap VG'' = \emptyset$.

Uma tal partição pode ser obtida colocando-se inicialmente todos os vértices em VG' e transferindo-se vértices de VG' para VG" em ordem topológica reversa até que a condição 2 de (1) seja satisfeita. Note que cada vértice transferido para VG" acrescenta de 0 a d arestas a aG(VG,VG") e nada a aG(VG",VG).

Sejam G'=G[VG'] e G"=G[VG"]. Seja A=aG-(aG'∪aG"). Note que {G',G",A} é uma partição de G.

Se A é "pequeno", o algoritmo é aplicado a G' para ocupar permanentemente os vértices de G' que são predecessores de algum vértice de G". Todas as pedras em G', exceto as permanentes, são removidas e o algoritmo é aplicado a G".

Se A é "grande", o algoritmo é aplicado a G". Sempre que o próximo vértice de G" a ser ocupado tiver predecessores em G', o algoritmo é aplicado a G' para ocupá-los e todas as pe-

dras em G', exceto as dos predecessores, são removidas. Após a ocupação do vértice de G", todas as pedras em G' são removidas e o algoritmo prossegue em G". Após a ocupação de G", todas as pedras em G" são removidas e o algoritmo é aplicado a G'.

Uma descrição mais detalhada de MELHOR-OCUPAÇÃO é dada a seguir. O parâmetro V é um conjunto de vértices de VG que não devem ser desocupados durante a atual chamada do algoritmo. A constante m_o será determinada mais adiante.

O algoritmo utiliza uma pilha auxiliar que armazena triplas (G',V',Z), onde $V'=V \cap VG'$ e $Z=\{({}_\iota G(\alpha), \phi G(\alpha)) : \alpha \in A\}$, criadas pelas chamadas recursivas do algoritmo.

MELHOR-OCUPAÇÃO (G,V)

Seja $m = |aG|$.

MO1. Se $m \leq m_o$, para todo $v \in VG$ em ordem topológica que não está ocupado faça

1. $U:=\{u : u$ não está ocupado e $(u,v) \in Z$ para alguma tripla (G',V',Z) na pilha$\}$;
2. para cada (G',V',Z) da pilha, se $VG' \cap U \neq \emptyset$ chame MELHOR-OCUPAÇÃO $(G', VG' \cap U)$;
3. ocupe v;
4. desocupe os vértices de U;
5. desocupe os vértices de VG-V cujos sucessores em G forem anteriores a v na ordem topológica ou o próprio v.

MO2. Se $m > m_o$, particione G em G',G" e A satisfazendo a condição (1).

Sejam $V' = V \cap VG'$ e $V'' = V \cap VG''$.

MO2.1. Corte Pequeno

Se $|A| \leq 2m/\lg m$, seja $U = \{{}_1G(\alpha) : \alpha \in A\}$.

Chame MELHOR-OCUPAÇÃO $(G', V' \cup U)$.

Chame MELHOR-OCUPAÇÃO (G'', V'').

Desocupe os vértices de $U - V'$.

MO2.2. Corte Grande

Se $|A| > 2m/\lg m$, seja $Z = \{({}_1G(\alpha), \phi G(\alpha)) : \alpha \in A\}$.

Empilhe (G', V', Z).

Chame MELHOR-OCUPAÇÃO (G'', V'').

Desempilhe (G', V', Z).

Chame MELHOR-OCUPAÇÃO (G', V').

□

Seja $G_o \in G(n,d)$. Mostraremos que a chamada MELHOR-OCUPAÇÃO $(G_o, \emptyset)$ com a pilha inicialmente vazia ocupa G_o.

Vamos fixar uma ordenação topológica $v_1, v_2, \ldots, v_n$ dos vértices de VG_o que será utilizada pelo algoritmo.

Um subgrafo H de G_o é *bom* se existem i,j $(1 \leq i \leq j \leq n)$ tais que $VH = \{v_i, v_{i+1}, \ldots, v_{j-1}, v_j\}$ e $H = G_o[VH]$. Neste caso, vamos denotar j por $j(H)$.

LEMA 1 - Seja $G \subseteq G_o$ bom e seja $V \subseteq VG$. Consideremos uma configuração inicial C de pedras no grafo. Consideremos, ainda, uma configuração inicial da pilha e seja $\mathcal{G}' = \{G' : (G', V', Z)$ está na pilha$\}$. Suponhamos que para todo (G', V', Z) na pilha:

1. G' é bom; (2)

2. $V' \subseteq VG'$; (3)

3. para todo $(u,v) \in Z$, $u \rightarrow_{G_o} v$, $u \in VG'$ e $j(H) < j(G_o[v])$ (4)

Suponhamos, ainda, que para todo $G_1 \in G' \cup \{G\}$

4. $\mathrm{Pred}G_o(VG_1) - C \subseteq VG_1 \cup \bigcup_{(G',V',Z) \text{ na pilha}} \{u \in VG' : v \in VG_o \text{ tq.} (u,v) \in Z\}$ (5)

Então a chamada MELHOR-OCUPAÇÃO (G,V) pára, deixando a pilha com sua configuração inicial, ocupa G (i.e., os vértices de VG-C) e os vértices de G_o que ficam acupados quando ela pára são os de $(C-VG) \cup V$.

DEMONSTRAÇÃO - Por indução sobre $(j(G), |aG|)$, considerando a ordem lexicográfica.

Se $(j(G), |aG|) = (1,0)$, o vértice v_1 é ocupado sem que a pilha seja usada e o teorema segue trivialmente.

Se $(j(G), |aG|) > (1,0)$, vamos considerar dois casos:

<u>Caso 1</u>: $|aG| \leq m_o$

Seja v o primeiro vértice de G em ordem topológica que não está ocupado (se existir).

Por (5), cada predecessor de v em G_o que não está ocupado está em VG' para algum G' em $\mathcal{G}'$. Um tal G' satisfaz $j(G') < j(G)$ (por (4)). Além disso, G', C, $VG' \cap U$ e a configuração inicial da pilha satisfazem as condições do lema. Por hipótese de indução, a chamada MELHOR-OCUPAÇÃO $(G', VG' \cap U)$ pára, deixando a pilha com sua configuração original e os vértices de $VG' \cap U$ ocupados.

Assim, as chamadas sucessivas a vários G' ocupam todos os predecessores de v em G_o que não estão em VG, deixando a pi-

lha com sua configuração inicial. Mais ainda, uma chamada não altera o resultado de uma anterior porque U permanece constante. O vértice v pode, então, ser ocupado.

Note que as remoções de pedras em G são feitas de tal maneira que imediatamente após a ocupação de um vértice de G os predecessores em G do próximo vértice de G a ser ocupado estão todos ocupados. Assim, os argumentos acima podem ser repetidos para cada vértice de G ocupado durante esta chamada.

Como VG é finito, a chamada pára, deixando a pilha com sua configuração inicial e ocupa G. A finitude de VG-V e a desocupação de seus vértices completam a demonstração.

Caso 2: $|aG| > m_o$

O grafo G é particionado em G', G'' e A satisfazendo (1). Note que $j(G'')=j(G)$ e a escolha de m_o, que será feita no Teorema 3, nos garantirá $j(G')<j(G)$ e $|aG''|< |aG|$. Além disso, G' e G'' são bons. Vamos considerar dois casos.

Caso 2.1: Corte Pequeno

O algoritmo é aplicado a argumentos $(G', V' \cup U)$, onde $V' = VG' \cap V$ e U é o conjunto dos vértices de $PredG(G'') \cap VG'$.

Por construção, $PredG_o(VG') \cap VG'' = \emptyset$. Portanto, G', $V' \cup U$, C e a configuração inicial da pilha satisfazem as condições do lema. Por hipótese de indução, a chamada MELHOR-OCUPAÇÃO $(G', V' \cup U)$ pára, deixando a pilha com sua configuração inicial, ocupa G' e os vértices de G_o que estão ocupados quando ela para são os de $(C-VG') \cup (V' \cup U)$.

Em seguida, o algoritmo é aplicado a (G'', V''), onde $V'' = VG'' \cap V$.

Como os vértices de $PredG_o(VG'') \cap VG' = U$ estão ocupados, G'', V'', $(C-VG') \cup (V' \cup U)$ e a configuração inicial da pilha satisfazem as condições do lema. Além disso, $|aG''| < |aG|$. Por hipótese de indução, a chamada MELHOR-OCUPAÇÃO (G'',V'') pára, deixando a pilha com sua configuração inicial, ocupa G'' e os vértices de G_o que estão ocupados quando ela termina são os de $(C-VG) \cup (V \cup U)$.

Assim, após as duas chamadas G foi ocupado e a pilha está com sua configuração inicial. A finitude de $U-V$ e a desocupação de seus vértices completam a demonstração.

Caso 2.2: Corte Grande

A tripla (G',V',Z) é empilhada e o algoritmo aplicado a (G'',V'').

Pelas discussões feitas no Caso 2.1, é fácil verificar que G'', V'', C e a configuração inicial da pilha satisfazem as condições do lema. Além disso, $|aG''| < |aG|$. Por hipótese de indução, a chamada MELHOR-OCUPAÇÃO (G'',V'') pára, deixando a pilha com a nova configuração, ocupa G'' e os vértices de G_o que estão ocupados quando ela termina são os de $(C-VG'') \cup V''$.

O desempilhamento de (G',V',Z) deixa a pilha com sua configuração inicial.

Em seguida, o algoritmo é aplicado a (G',V'). Analogamente ao caso 2.1, podemos aplicar a hipótese de indução a esta aplicação. Portanto, a chamada MELHOR-OCUPAÇÃO (G',V') pára, deixando a pilha com sua configuração inicial, ocupa G' e os vértices de G_o que estão ocupados quando ela termina são os de $(C-VG) \cup V$.

□

TEOREMA 2 - A chamada MELHOR-OCUPAÇÃO $(G_o,\emptyset)$ com a pilha vazia e configuração inicial $C=\emptyset$ ocupa G_o.

DEMONSTRAÇÃO - Segue do Lema 1.

□

Vamos, agora, analisar a complexidade do algoritmo.

TEOREMA 3 - Seja $d\geq 2$. Então existe uma constante c tal que MELHOR-OCUPAÇÃO ocupa qualquer grafo acíclico com m arestas e grau de entrada máximo d em espaço $cm/\ell gm$.

DEMONSTRAÇÃO - Seja q(m) o maior espaço utilizado por MELHOR-OCUPAÇÃO para grafos com m arestas e grau de entrada máximo d.

Seja m_o o menor inteiro tal que $\ell g\ m_o \geq 12$, $m_o \geq 16d$ e $d \leq m_o/2(\ell g\ m_o)^2$. Seja $c=2\ell g\ m_o$. Vamos mostrar, por indução sobre m, que $q(m) \leq cm/\ell gm$.

Se $m \leq m_o$, os vértices do grafo são ocupados em ordem topológica. Sempre que um vértice não tiver sucessores a serem ocupados, ele é desocupado e assim permanece até o final do algoritmo. Assim, o número de pedras em G em cada instante da estratégia nunca excede m+1.

Portanto,

$$q(m) \leq m+1 \leq 2m = (c/\ell g\ m_o)m$$

$$\leq cm/\ell g\ m$$

Se $m \geq m_o$, vamos considerar dois casos.

<u>Caso 1</u>: Corte Pequeno

$$q(m) \leq q(m/2 + d + m/\ell gm) + 2m/\ell gm$$

$$\leq \frac{c(m/2 + d + m/\ell gm)}{\ell g(m/2)} + \frac{2m}{\ell g(m/2)}$$

$$= \frac{cm(1/2+d/m+1/\ell gm+2/c)}{\ell gm - 1}$$

Como $\ell gm \geq \ell gm_o \geq 12$ e $m \geq m_o \geq 16d$, temos

$$q(m) \leq \frac{cm}{\ell gm-1} (1/2+1/16+1/12+1/12)$$

$$= \frac{cm}{\ell gm-1} (35/48)$$

$$\leq cm \frac{3/4}{\ell gm-1}$$

$$\leq cm/\ell gm$$

<u>Caso 2</u>: Corte Grande

Neste caso,

$$|aG''| \leq m/2 + m/\ell gm + d - |A|$$

$$\leq m/2 - m/\ell gm + d$$

Logo,

$$q(m) \leq q(m/2+d-m/\ell gm)+q(m/2-m/\ell gm)+d$$

$$\leq \frac{c(m/2+d-m/\ell gm)}{\ell g(m/2-m/\ell gm)} + \frac{c(m/2-m/\ell gm)}{\ell g(m/2-m/\ell gm)} + d$$

$$\leq \frac{c(m+d-2m/\ell gm)+d\ \ell gm}{\ell gm+\ell g(1/2-1/\ell gm)}$$

$$\leq \frac{cm(1+d/m+d\ \ell gm/cm-2/\ell gm)}{\ell gm + \ell g(1/2-1/\ell gm)}$$

Como $2\,\ell gm \geq c$ temos que $d/m \leq 2d\,\ell gm/cm$ e, portanto,

$$d/m + d\,\ell gm/cm \leq (3/c)\,d\,\ell gm/m \leq d\,\ell gm/m$$

De $d \leq m_o/2\,(\ell gm_o)^2 \leq m/2(\ell gm)^2$ segue que

$$d\,\ell gm/m - 2/\ell gm \leq -3/2\ell gm$$

Além disso,

$$\ell g(1/2 - 1/\ell gm) \geq \ell g(1/2-1/12)=\ell g(5/12) \geq -3/2$$

Portanto,

$$q(m) \leq \frac{cm(1-3/2\ell gm)}{\ell gm - 3/2} = cm/\ell gm$$

□

COROLÁRIO 4 - Seja $d \geq 2$. Então para todo $n \geq 1$, $Ed(n) \leq cdn/\ell gn$, onde c é a constante do Teorema 3.

DEMONSTRAÇÃO - Para todo $n \geq 1$,

$$E_d(n) \leq q(dn) \leq cdn/\ell g(dn) \leq cdn/\ell gn$$

□

3 - UM LIMITE INFERIOR PARA $E_d(n)$

Para mostrar que o algoritmo MELHOR-OCUPAÇÃO é ótimo a menos de um fator constante, vamos construir uma família de grafos cuja ocupação requer espaço $\Omega(n/\ell gn)$.

Inicialmente, vamos construir uma família $\{G(k)\}_{k \geq 5}$ a

partir da família $\{\bar{S}_N\}_{N\geq 1}$ de N-superconcentradores do Teorema 1.2.13.

1. $G(5) = \bar{S}_{32}$

2. para $k\geq 5$, $G(k+1)$ é composto de um conjunto de fontes $\{x_j^{k+1}: 1 \leq j \leq 2^{k+1}\}$, um conjunto de sorvedouros $\{y_j^{k+1}: 1 \leq j \leq 2^{k+1}\}$, duas cópias S^1 e S^2 de S_{2^k} e duas cópias G^1 e G^2 de $G(k)$, ligados como na Figura 2.1.

É imediato que $|VG(5)| = 190\times 2^5$ e para todo $k\geq 5$,

$$|VG(k+1)| = 2(|VG(k)| + 189\times 2^k + 2^{k+1}) =$$

$$= 2|VG(k)| + 191\times 2^{k+1}$$

Resolvendo a recorrência obtemos

$$|VG(k)| = (190+191(k-5))2^k \text{ , para todo } k\geq 5$$

Para analisar o espaço necessário para ocupação destes grafos, vamos utilizar constantes $c_1, c_2, c_3, c_4 > 0$ tais que

(D0) $32c_1,\ 32c_2,\ 32c_3 \in \mathbb{N}$ e $c_4 = 1/32$

(D1) $c_1,\ c_2,\ c_3 < 1$

(D2) $4c_2 < c_3$

(D3) $c_3 \leq 1 - c_2$

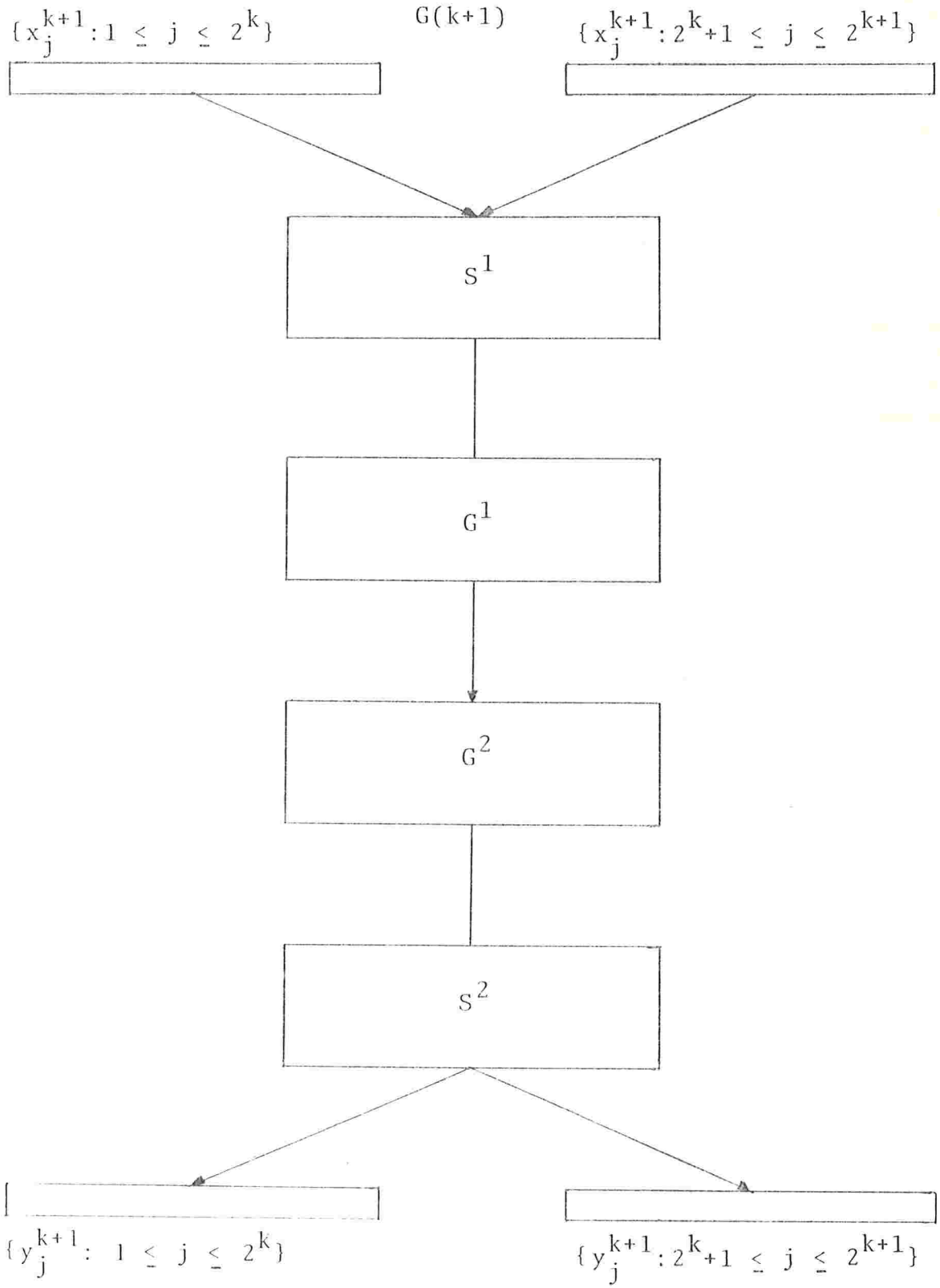
(D4) $2c_4 \leq c_2$

(D5) $4c_2 < c_1$

(D6) $c_1 \leq 1 - 2c_2$

(D7) $2c_1 \leq c_3$

(d8) $c_1 \leq c_3/2 - c_2$



——— indica identificação de vértices correspondentes

⟶ indica arestas ligando vértices correspondentes

Figura 2.1

Um exemplo de constantes que satisfazem estas desigualdades é $c_1 = 9/32$, $c_2 = 2/32$, $c_3 = 22/32$, $c_4 = 1/32$.

O lema abaixo estabelece um limite inferior para o espaço necessário para a ocupação de G(k). Sua demonstração consiste, basicamente, em aplicar o Lema Fundamental a subgrafos de G(k). Nela suporemos, sem perda de generalidade, que uma estratégia que ocupa G(k) durante um intervalo [1,T] satisfaz a condição

$$\text{para todo } t \in [1,T-1],\ C_t \neq \emptyset \qquad (1)$$

Caso contrário, para todo $t \in [1,T-1]$ tal que $C_t=\emptyset$, existem $u, v \in VG(k)$ tais que u é desocupado no instante t e v é ocupado no instante t+1; invertendo a ordem destes dois lances, para todo t, obtemos uma nova estratégia que ocupa G(k) no mesmo espaço e no mesmo tempo de estratégia original e que satisfaz (1).

LEMA 1 - Para todo $k \geq 5$, se pelo menos $c_1\ 2^k$ sorvedouros de G(k) são ocupados durante um intervalo $T = [t_1, t_2]$ tal que $|C_{t_1-1}| \leq c_2\ 2^k$, então existe um sub-intervalo T' de T tal que pelo menos $c_3\ 2^k$ fontes de G(k) são ocupadas durante T' e para todo $t \in T'$, $|C_t| \geq c_4\ 2^k$.

DEMONSTRAÇÃO - Por indução sobre k.

Suponhamos que 13 sorvedouros de G(5) são ocupados durante um intervalo T a partir de uma configuração de 3 pedras no grafo. Seja T' o menor sub-intervalo de T durante o qual os primeiros 4 sorvedouros são ocupados. Pelo Lema Fundamental , pelo menos $32-3 = 29 > 26$ fontes de G(5) são ocupados durante

T'. Além disso, em cada instante de T' há pelo menos uma pedra em $G(5)$.

Suponhamos a tese válida para k e consideremos um intervalo $T = [t_1, t_2]$ tal que pelo menos $c_1\ 2^{k+1}$ sorvedouros de $G(k+1)$ são ocupados durante T e $|C_{t_1-1}| \leq c_2\ 2^{k+1}$.

Vamos considerar 4 casos.

Caso 1: Existe um intervalo $[t_3, t_4] \subseteq T$ tal que pelo menos $c_3\ 2^{k-1}$ fontes de G^1 são ocupadas durante $[t_3, t_4]$ e para todo $t \in [t_3, t_4]$, $|C_t| \geq c_2\ 2^k$.

É fácil ver que os subgrafos $G(k+1)[VS^1 \cup \{x_j^{k+1} : 1 \leq j \leq 2^k\}]$ e $G(k+1)\ [VS^1 \cup \{x_j^{k+1} : 2^k+1 \leq j \leq 2^{k+1}\}]$ de $G(k+1)$ são 2^k-super concentradores.

Seja t_5 o último instante anterior a t_3 tal que $|C_{t_5}| \leq$ $\leq c_2\ 2^{k+1}$. Durante $[t_5+1, t_4]$ pelo menos $c_3\ 2^{k-1} > c_2\ 2^{k+1}$ (por (D2)) fontes de G^1 (sorvedouros de S^1) são ocupadas. Assim, aplicando o Lema Fundamental aos dois subgrafos acima e utilizando a desigualdade (D3) temos que durante $[t_5+1, t_4]$ pelo menos $2(2^k - c_2\ 2^{k+1}) = (1-2c_2)2^{k+1} \geq c_3\ 2^{k+1}$ fontes de $G(k+1)$ são ocupadas. Além disso, para todo $t \in [t_5+1, t_4]$, $|C_t| \geq c_2\ 2^k \geq$ $\geq c_4\ 2^{k+1}$ (por (D4)).

Caso 2: Existe um intervalo $[t_3, t_4] \subseteq T$ tal que pelo menos $c_3\ 2^{k-1}$ fontes de G^2 são ocupadas durante $[t_3, t_4]$ e para todo $t \in [t_3, t_4]$, $|C_t| \geq c_2\ 2^k$.

É fácil ver que para todo $i \geq 5$, existe um 2^i-fluxo que liga as fontes de $G(i)$ a seus sorvedouros.

O subgrafo de $G(k+1)$ gerado pelos vértices de $S^1, \{x_j^{k+1}$:

$: 1 \leq j \leq 2^k\}$, os vértices de um 2^k-fluxo que liga as fontes de G^1 a seus sorvedouros e as fontes de G_2 é um 2^k-superconcentrador. Também o subgrafo gerado pelos vértices de S^1, $\{x_j^{k+1}: : 2^k + 1 \leq j \leq 2^{k+1}\}$, os vértices de um 2^k-fluxo que liga as fontes de G^1 a seus sorvedouros e as fontes de G_2^2.

O resto da demonstração deste caso é análoga à do Caso 1.

Caso 3: Existe um intervalo $[t_3, t_4] \subseteq T$ tal que pelo menos $c_1\ 2^k$ sorvedouros de $G(k+1)$ são ocupados durante $[t_3, t_4]$ e para todo $t \in [t_3, t_4]$, $|C_t| \geq c_2\ 2^k$.

Durante $[t_3, t_4]$ $c_1\ 2^{k-1}$ sorvedouros de $\{y_j^{k+1}: 1 \leq j \leq 2^k\}$ ou $c_1\ 2^{k-1}$ sorvedouros de $\{y_j^{k+1}: 2^k+1 \leq j \leq 2^{k+1}$ são ocupados. Suponhamos, sem perda de generalidade, que ocorre o primeiro.

O subgrafo de $G(k+1)$ gerado pelos vértices de S^1 e S^2, os de um 2^k-fluxo que liga as fontes dos sorvedouros de G^1 e outro em G^2, $\{x_j^{k+1}: 1 \leq j \leq 2^k\}$ e $\{y_j^{k+1} : 1 \leq j \leq 2^k\}$ é um 2^k-superconcentrador. Também o subgrafo gerado pelos vértices de S^1 e S^2, os de um 2^k-fluxo que liga as fontes aos sorvedouros de G^1 e outro em G^2, $\{x_j^{k+1}: 2^k+1 \leq j \leq 2^{k+1}\}$ e $\{y_j^{k+1}: 1 \leq j \leq 2^k\}$.

O resto da demonstração deste caso é análoga à do Caso 1, utilizando-se as desigualdades (D3), (D4) e (D5).

Caso 4: Nenhum dos casos anteriores ocorre.

Como o Caso 3 não ocorre, existe $t_3 \in T$ tal que menos de $c_1\ 2^k$ sorvedouros de $G(k+1)$ são ocupados durante $[t_1, t_3]$ e $|C_{t_3}| < c_2\ 2^k$. Senão, seja t_o o instante em que o $(c_1\ 2^k)$-ésimo sorvedouro é ocupado. Para todo $t \in [t_1, t_o-1]$, $t \geq c_2 2^k$. Como no instante t_o uma pedra é colocada no grafo, o intervalo $[t_1, t_o]$ satisfaz as condições do Caso 3.

Durante $[t_3+1, t_2]$ pelo menos $c_1\ 2^k$ sorvedouros de $G(k+1)$ são ocupados. Portanto, durante $[t_3+1, t_2]$ $c_1\ 2^{k-1}$ sorvedouros de $\{y_j^{k+1}: 1 \leq j \leq 2^k\}$ ou $c_1\ 2^{k-1}$ sorvedouros de $\{y_1^{k+1}: 2^k+1 \leq j \leq 2^{k+1}\}$ são ocupados. Suponhamos, sem perda de generalidade, que ocorre o primeiro.

O subgrafo $G(k+1)\ [VS^2 \cup \{y_j^{k+1}: 1 \leq j \leq 2^k\}]$ de $G(k+1)$ é um 2^k-superconcentrador. Durante $[t_3+1, t_2]$ pelo menos $c_1\ 2^{k-1} > c_2\ 2^k$ (por (D5)) sorvedouros deste subgrafo são ocupados. Como $|VG^2 \cap C_{t_3}| \leq c_2\ 2^k$ segue do Lema Fundamental que durante $[t_3+1, t_2]$, pelo menos $(1-c_2)2^k \geq c_1\ 2^k$ (por (D6)) fontes de S^2 (que são os sorvedouros de G^2) são ocupados. Pela hipótese de indução, existe $[t_4, t_5] \subseteq [t_3+1, t_2]$ tal que pelo menos $c_3\ 2^k$ fontes de G^2 são ocupadas durante $[t_4, t_5]$ e para todo $t \in [t_4, t_5]$, $|VG^2 \cap C_t| \geq c_4\ 2^k$.

Como o Caso 2 não ocorre, existe $t_6 \in [t_4, t_5]$ tal que menos de $c_3\ 2^{k-1}$ fontes de G^2 são ocupadas durante $[t_4, t_6]$ e $|C_{t_6}| < c_2\ 2^k$.

Durante (t_6+1, t_5) pelo menos $(c_3/2)2^k - c_2\ 2^k \geq c_1\ 2^k$ (por (D8)) sorvedouros de G^1 são ocupados e $|VG^1 \cap C_{t_6}| < c_2\ 2^k$. Pela hipótese de indução, existe $[t_7, t_8] \subseteq [t_6+1, t_5]$ tal que pelo menos $c_3\ 2^k$ fontes de G^1 (que são os sorvedouros de S^1) são ocupados durante $[t_7, t_8]$ e para todo $t \in [t_7, t_8]$, $|VG' \cap C_t| \geq c_4\ 2^k$.

Como o Caso 1 não ocorre, existe $t_9 \in [t_7, t_8]$ tal que menos de $c_3\ 2^{k-1}$ fontes de G^1 são ocupadas durante $[t_9, t_7]$ e $|C_{t_9}| < c_2\ 2^k$.

Durante $[t_9+1, t_8]$ pelo menos $c_3\ 2^{k-1} > c_2\ 2^k$ (por (D2)) sorvedouros de S^1 são ocupados. Como $|VS^1 \cap C_{t_9}| \leq c_2\ 2^k$, segue da aplicação do Lema Fundamental aos dois subgrafos do Caso 1, que durante $[t_9+1, t_8]$ pelo menos $2(1-c_2)2^k = (1-c_2)2^{k+1} \geq$ $\geq c_3\ 2^{k+1}$ (por (D3)) fontes de $G(k+1)$ são ocupadas. Além disso, para todo $t \in [t_9+1, t_8] \subseteq [t_7, t_8] \subseteq [t_4, t_5]$, $|C_t| \geq$ $\geq |VG^1 \cap C_t| + |VG^2 \cap C_t| \geq c_4\ 2^{k+1}$, já que $VG^1 \cap VG^2 = \emptyset$.

TEOREMA 2 - Para todo $n \geq 2$, existe $G_n \in G(n,2)$ cuja ocupação requer espaço $\Omega(n/\ell gn)$.

DEMONSTRAÇÃO - Para $n < |VG(5)|$, o teorema é válido pois a ocupação de um grafo qualquer de $G(n,d)$ requer espaço $1 \geq$ $\geq (1/|VG(5)|)n/\ell gn$.

Para $n \geq |VG(5)|$, seja k tal que $|VG(k)| \leq n < |VG(k+1)|$. Seja G_n o grafo obtido pelo acréscimo de $n-|VG(k)|$ vértices a $G(k)$. É imediato que $G_n \in G(n,2)$. Pelo Lema 1, a ocupação de G_n requer espaço $c_4\ 2^k$.

Para todo $k \geq 5$, temos

$$|VG(k+1)| = (190+191(k-4))2^{k+1}$$

$$\leq 5\ |VG(k)|$$

Assim, $n/5 \leq |VG(k)| \leq n$, i.e.,

$$n/5 \leq (190 + 191(k-5))2^k \leq n$$

É fácil ver que $k \leq \ell gn$. Portanto,

$$c_4\ 2^k \geq \frac{c_4 n}{5(190+191(k-5))} \geq \frac{c_4}{5(191\ k)}\ n$$

$$\geq \frac{c_4}{905} \cdot \frac{n}{\lg n}$$

Logo, escolhendo uma constante suficientemente pequena obtemos o teorema.

□

COROLÁRIO 3 - Para todo $d \geq 2$. $E_d(n) = \Omega(n/\lg n)$.

DEMONSTRAÇÃO - Segue de $E_d(n) \geq E_2(n)$, para todo $n \geq 1$ e do Teorema 2.

□

COROLÁRIO 4 - Para todo $d \geq 2$, $E_d(n) = \theta(n/\lg n)$.

DEMONSTRAÇÃO - Segue do Corolário 3 e do Corolário 2.2.4.

□

CAPITULO 3

TEMPO vs ESPAÇO NO JOGO DAS PEDRAS

1 - INTRODUÇÃO

No Capítulo 2 vimos que qualquer grafo de $G(n,d)$ pode ser ocupado em espaço E se $\Omega(n/\ell gn) \leq E \leq n$. No entanto, nenhuma consideração foi feita a respeito do tempo necessário para esta ocupação.

É imediato que quanto menor o espaço disponível, maior o tempo necessário para a ocupação. Surge, então, a pergunta de como o tempo T aumenta à medida que o espaço disponível E diminui.

Na Seção 2 apresentamos um algoritmo que ocupa qualquer grafo de $G(n,d)$ em espaço $E \geq \Omega(n/\ell gn)$ em tempo

$$T \leq O(E\ 2^{2^{(d+1)n/E)}}).$$

Na Seção 3 construimos uma família de grafos cuja ocupação em espaço $E \leq O(n)$ requer tempo

$$T \geq \Omega(E\ 2^{2^{\Omega(n/E)}})$$

Usaremos a versão do Lema Fundamental do Lema 1.2.3.

2 - UM LIMITE SUPERIOR

Nesta seção apresentamos um algoritmo que chamaremos OCUPAÇÃO-RÁPIDA que ocupa eficientemente qualquer grafo de $G(n,d)$ em espaço (pré-fixado) $E = \Omega(n/\ell gn)$.

A análise do algoritmo nos leva a um limite superior da forma

$$T \leq E(c_7 d)^{c_8^{(d+1)n/E}}$$

para o tempo utilizado pelo mesmo para ocupar qualquer grafo de $G(n,d)$ em espaço E, onde c_7, c_8 são constantes positivas.

O algoritmo e sua análise dependem de constantes positivas que satisfazem as seguintes desigualdades:

(D1) $c_3 < 2c_2$

(D2) $2c_2 < c_1$

(D3) $c_1 + c_2 < 1/2$

(D4) $c_3 < \dfrac{1 - 2c_1 - 2c_2}{2(1-c_1)}$

(D5) $\dfrac{2}{(1-2c_1) - 2c_3(1-c_1) - 2c_2} \leq c_4$

(D6) $$-c_3c_5 \leq \ell g(1/2-c_1)$$

(D7) $$3/2c_2 \leq c_4$$

(D8) $$(1+\ell g\, c_4)\left(\frac{6}{c_4(2c_2-c_3)}\right)^{1/2} \leq c_5$$

(D9) $$1 + \ell g c_4 \leq c_5$$

(D10) $$3c_6 \leq (2c_2-c_3) - 3\,\frac{1+\ell g c4}{c_4c_5}$$

(D11) $$1 + 1/c_6 \leq c_7^{(c_8^{c_3}-2)c_8^{1-c_3}}$$

(D12) $$1 < c_7, c_8$$

(D13) $$1 < \left(1 - \frac{2}{c_8^{c_3}}\right) c_8(\ell n c_7)(\ell n c_8)$$

(D14) $$6 < c_4$$

(D15) $$5 < c_2c_5$$

Um exemplo de constantes que satisfazem estas desigualdades é:

$c_1 = 11/50$, $c_2 = 21/200$, $c_3 = 1/5$, $c_4 = 53$, $c_5 = 400$,

$c_6 = 1/343, \quad c_7 = 29, \quad c_8 = 40.$

O algoritmo OCUPAÇÃO-RÁPIDA é baseado em MELHOR-OCUPAÇÃO, com três diferenças. Uma delas é o ponto onde o corte é feito. Outra é no Corte Grande, onde o algoritmo utiliza a função

$$p(d,m,E) = \left\lceil \frac{(2c_2-c_3)E^2 - 2(d+1)E}{dm} \right\rceil - 1$$

para $d \geq 2$, $m \geq 1$ e $E \geq 1$ e que será denotada simplesmente por p. A recursão é feita sobre o tamanho do grafo.

A seguir descrevemos informalmente o algoritmo que ocupa $G \in \mathcal{G}(n,d)$ com E pedras.

OCUPAÇÃO-RÁPIDA (G,E)

Seja $m = |VG| + |aG|$.

OR1. Se $m \leq E$, ocupe G em ordem topológica dos vértices, sem remoções.

OR2. Se $m > E$, particione G em dois subgrafos G' e G'' e um conjunto A de arestas (como em MELHOR-OCUPAÇÃO) tais que

$$\lceil m/2 \rceil - \lfloor c_2 E \rfloor - d \leq |aG'| + |VG'| \leq \lceil m/2 \rceil - \lfloor c_2 E \rfloor$$

OR2.1. Corte Pequeno

Se $|A| \leq \lfloor c_1 E \rfloor$, particione as E pedras em dois conjuntos P' e P'' com $\lceil (1-c_1)E \rceil$ e $\lfloor c_1 E \rfloor$ pedras, respectivamente. Chame OCUPAÇÃO-RÁPIDA $(G', \lceil (1-c_1)E \rceil)$ usando as pedras de P'; durante esta chamada, sempre que um predecessor de algum vértice de G'' tiver que ser ocupado, utilize uma pedra de P'' para fazê-lo e não a remova até o fim da chamada. Remova as pedras de P' de G' e utilize-as para chamar OCUPAÇÃO-RÁPIDA$(G'', \lceil (1-c_1)E \rceil)$.

OR2.2. Corte Grande

Se $|A| > \lfloor c_1 E \rfloor$, particione as E pedras em dois conjuntos P' e P'' de $\lfloor (E-dp)/2 \rfloor$ pedras cada e um conjunto P de $E-2\lfloor (E-dp)/2 \rfloor \geq dp$ pedras. Chame OCUPAÇÃO-RÁPIDA$(G'', \lfloor (E-dp)/2 \rfloor)$ usando as pedras de P''. Sempre que o próximo vértice de G'' a ser ocupado tiver predecessores não-ocupados em G', suspenda temporariamente a ocupação de G''. Chame OCUPAÇÃO-RÁPIDA$(G', \lfloor (E-dp)/2 \rfloor)$ usando as pedras de P'; durante esta chamada, use as pedras de P para ocupar os (não mais que dp) predecessores dos próximos p vértices de G'' a serem ocupados e não as remova até que os p vértices sejam ocupados. Em seguida, prossiga a ocupação de G'', removendo todas as pedras de G' após a ocupação dos p vértices.

□

Antes de iniciarmos a análise do algoritmo, vamos introduzir algumas notações.

Se OCUPAÇÃO-RÁPIDA for chamado com argumentos (G,E), onde $|VG|+|aG|=m$, diremos que OCUPAÇÃO-RÁPIDA foi chamado com *argumentos de tamanho* (m,E). Durante sua execução, o algoritmo chama a si próprio para ocupar grafos G' e G''. Denotaremos o tamanho do argumento correspondente por (m',E'). Assim, m' pode ser $|VG'|+|aG'|$ ou $|VG''|+|aG''|$ e E' pode ser $\lceil(1-c_1)E\rceil$ ou $\lfloor(E-dp)/2\rfloor$.

Seja $m_o = c_4 d$. O número m_o indica o ponto a partir do qual o algoritmo torna-se não-trivial, como veremos mais adiante.

LEMA 1: Consideremos uma chamada de OCUPAÇÃO-RÁPIDA com argumentos de tamanho (m,E). Suponhamos que $m > E \geq 1$, $m > m_o$ e, se ocorrer o Corte Grande, que $p \geq 1$. Então para toda chamada recursiva do algoritmo com argumentos de tamanho (m',E'), $E' \geq 1$ e $m'/E' \leq m/E - c_3$.

DEMONSTRAÇÃO - Vamos considerar dois casos.

Caso 1: Corte Pequeno

Da definição do algoritmo segue que na chamada em G'

$$\frac{m'}{E'} \leq \frac{\lfloor m/2\rfloor-\lfloor c_2E\rfloor}{\lceil(1-c_1)E\rceil} \leq \frac{(m/2+1)-(c_2E-1)}{(1-c_1)E} \leq \frac{m/2+c_2E+d}{(1-c_1)E}$$

e na chamada em G''

$$\frac{m'}{E'} \leq \frac{\lfloor m/2 \rfloor + \lfloor c_2 E \rfloor + d}{\lceil (1-c_1)E \rceil} \leq \frac{m/2 + c_2 E + d}{(1-c_1)E}$$

Utilizando (D5) temos

$$m \geq m_o = c_4 d \geq \frac{2d}{(1-2c_1) - [2c_3(1-c_1) + 2c_2]}$$

e, como $m \geq E$,

$$2d \leq (1-2c_1)m - [2c_3(1-c_1) + 2c_2]E$$

Rearranjando os termos da desigualdade e somando m aos dois lados da mesma obtemos

$$m + 2c_2E + 2d \leq 2(1-c_1)m - 2c_3(1-c_1)E$$

i.e.,

$$\frac{m}{2} + c_2E + d \leq \left(\frac{m}{E} - c_3\right)(1-c_1)E$$

Portanto, $m'/E' \leq m/E - c_3$.

Além disso, como $E \geq 1$ e $1-c_1 > 0$ (por (D3)) temos

$$E' = \lceil (1-c_1)E \rceil \geq 1.$$

<u>Caso 2</u>: Corte Grande

Pela definição do algoritmo,

$$E' \geq \frac{E - dp - 1}{2}$$

e

$$m' \leq \max \{\lfloor m/2 \rfloor + \lfloor c_2 E \rfloor + d - (\lfloor c_1 E \rfloor + 1), \lceil m/2 \rceil - \lfloor c_2 E \rfloor\}$$

Utilizando (D2) obtemos

$$\lfloor m/2 \rfloor + \lfloor c_2 E \rfloor + d - (\lfloor c_1 E \rfloor + 1)$$

$$\leq m/2 + c_2 E + d - c_1 E$$

$$\leq m/2 + (c_2 - c_1) E + d$$

$$\leq m/2 - c_2 E + (d+1)$$

Por outro lado,

$$\lceil m/2 \rceil - \lfloor c_2 E \rfloor \leq m/2 - c_2 E + 2$$

$$\leq m/2 - c_2 E + (d+1)$$

Assim, durante a atual chamada do algoritmo sempre teremos

$$m' \leq m/2 - c_2 E + (d+1)$$

Da definição de p segue que

$$p \leq \frac{E((2c_2 - c_3)E - 2(d+1))}{dm} - 1$$

Rearranjando os termos da desigualdade temos

$$2(d+1) - 2c_2E \leq -(m/E)(dp+d) - c_3E$$

$$\leq -(m/E)(dp+1) - c_3E$$

e somando m a cada lado,

$$m + 2(d+1) - 2c_2E \leq m - (m/E)\ (dp+1) - c_3E$$

Como $p \geq 1$,

$$m - 2c_2E + 2(d+1) \leq m - (m/E)(dp+1) - c_3(E-dp-1)$$

i.e.,

$$m - 2c_2E + 2(d+1) \leq (m/E - c_3)(E-dp-1)$$

Portanto, $m'/E' \leq m/E - c_3$.

Para mostrar que $E' \geq 1$ basta provar que $dp \leq E-2$. De fato, utilizando (D1) e (D3)

$$dp + 1 \leq d\left(\left\lfloor \frac{(2c_2-c_3)E^2-2(d+1)E}{dm} \right\rfloor - 1\right) + 1$$

$$\leq \frac{(2c_2-c_3)E^2 - 2(d+1)E}{m}$$

$$= E((2c_2-c_3)(E/m) - 2(d+1)/m)$$

$$\leq E(2c_2-c_3-2(d+1)/m)$$

$$\leq E(2c_2-c_3)$$

$$< E$$

i.e.,

$$dp \leq E-2$$

□

O Lema 1 nos garante que, sob certas condições, a razão m/E tem um declínio mínimo constante à medida em que descemos na recursão.

LEMA 2 - Seja (m,E) tal que $E \geq c_5(\ell gd)m/\ell gm$. Então

1. se $m \leq m_o$ então $E \geq m$ (i.e., problemas de tamanho (m,E) não envolvem chamadas recursivas);

2. se $m > m_o$ e $E < m$ então $p \geq 1$;

3. se $m > m_o$ e $E < m$ então para toda chamada recursiva com argumentos de tamanho (m',E') temos $E' \geq c_5(\ell gd)m'/\ell gm'$, i.e., a condição sobre E e m da hipótese do lema é hereditária.

DEMONSTRAÇÃO É puramente técnica.

1. De (D9), $d \geq 2$ e $m \leq m_o = c_4 d$ segue que

$$E \geq \frac{c_5 \ell gd}{\ell gm} m \geq \frac{(1+\ell gc_4)\ell gd}{\ell gm_o} m = \frac{(1+\ell gc_4)\ell gd}{\ell gc_4+\ell gd} m \geq m$$

2. De (D8) segue que

$$\frac{c_5 \ell g d}{\ell g m_o} \geq \frac{c_5}{1 + \ell g c_4} \geq \left(\frac{6}{c_4(2c_2 - c_3)} \right)^{1/2}$$

Por (D14), $m_o \geq 12$ e, portanto,

$$6 \leq \frac{(c_5 \ell g d)^2}{(\ell g m_o)^2} c_4 \ (2c_2 - c_3)$$

$$= (2c_2 - c_3)(c_5 \ell g d)^2 \ \frac{m_o/d}{(\ell g m)^2}$$

$$\leq (2c - c_3) \ \frac{(c_5 \ell g d)^2}{d} \ \frac{m}{(\ell g m)^2}$$

Da hipótese de que $E \geq c_5(\ell g d)m/\ell g^m$ segue que

$$\frac{(2c_2 - c_3)E^2}{dm} \geq 6$$

e de $E \leq m$ e $d \geq 2$ segue que

$$p \geq \frac{(2c_2 - c_3)E^2}{dm} - \frac{2(d+1)}{d} (E/m) - 2 \geq 6 - 3 - 2 = 1$$

3. Seja $m_1 = m/2 - c_2 E - d$. Vamos considerar dois casos.

<u>Caso 1</u>: $m \geq m_1$

De (D7) segue

$$m \geq m_o = c_4 d \geq \frac{3}{2} \frac{d}{c_2} \geq \frac{(d+1)}{d} \frac{d}{c_2} = \frac{d+1}{c_2}$$

Da hipótese de que $E \leq m$ e de (D2) segue que

$$m' \geq m \left(\frac{1}{2} - \frac{c_2 E}{m} - \frac{d}{m}\right)$$

$$\geq m\left(\frac{1}{2} + (c_2 - c_1)\frac{E}{m} - \frac{(d+1)}{m}\right)$$

$$\geq m\left(\frac{1}{2} + c_2 - c_1 - \frac{(d+1)}{m}\right)$$

$$\geq m\left(\frac{1}{2} - c_1\right)$$

Utilizando (D6) temos

$$\ell g m' \geq \ell g m + \ell g\,(1/2 - c_1) \geq \ell g m - c_3\, c_5\, \ell g d$$

Pelo Lema 1 e pela parte 2 deste lema

$$m'/E' \leq m/E - c_3$$

Da hipótese de que $E \geq c_5(\ell g d)m/\ell g m$ segue que

$$\frac{m'}{E'} \leq \frac{m}{E} - c_3 \leq \frac{\ell g m}{c_5\, \ell g d} - c_3$$

$$= \frac{1}{c_5 \ell g d}\,(\ell g m - c_3 c_5\, \ell g d)$$

$$\leq \frac{\ell g m'}{c_5 \ell g d}$$

Logo, $E' \geq c_5(\ell gd)m'/\ell gm$.

Caso 2: $m < m_1$.

Neste caso, m' é o tamanho de G''. Vamos novamente considerar dois casos.

Caso 2.1: Corte Pequeno

De $E < m$ e da desigualdade (D3) segue que:

$$m' \geq \lfloor m/2 \rfloor + \lfloor c_2E \rfloor - \lfloor c_1E \rfloor$$

$$\geq (m/2-1) + (c_2E-1) - c_1E$$

$$\geq m/2 - (c_1-c_2)E - 2$$

$$> (1/2 - c_1+c_2)E - 2$$

$$> 2c_2E - 2$$

Como $E \geq c_5(\ell gd)\ m/\ell gm$,

$$m' > \frac{2\ c_2c_5(\ell gd)m}{\ell gm} - 2$$

$$\geq 2\ c_2c_5 - 2$$

$$> 8 \qquad \text{(por (D15))}$$

Portanto, $m_1/\ell g m_1 \geq m'/\ell g m'$. Pela demonstração do Caso 1,

$$E' \geq c_5(\ell g d)\frac{m_1}{\ell g m_1} \geq c_5(\ell g\ d)\frac{m'}{\ell g m'}$$

<u>Caso 2.2</u>: Corte Grande

Neste caso,

$$m' + |A| \geq \lfloor m/2 \rfloor + \lfloor c_2 E \rfloor \geq m/2 + c_2 E - 2$$

Como o grau de entrada máximo do grafo é d temos

$$m' \geq \frac{1}{d+1}(m/2 + c_2 E - 2)$$

De $E \geq c_5(\ell g d)m/\ell g m$, $m \geq m_o$ e das desigualdades (D14) e (D15) segue que

$$m' \geq \frac{1}{d+1}\left(\frac{m_o}{2} + \frac{c_2 c_5(\ell g d)m}{\ell g m} - 2\right)$$

$$\geq \frac{1}{d+1}\left(\frac{c_4\ d}{2} + c_2 c_5 - 2\right)$$

$$\geq \frac{1}{d+1}(3d + 5 - 2)$$

$$= 3$$

O resto da demonstração é análoga à do Caso 2.1. □

O Lema 2 nos dá a correção do algoritmo.

Seja T(G,E) o tempo gasto por OCUPAÇÃO-RÁPIDA para ocupar o grafo G em espaço $E \geq c_5\ (\ell gd)\ m/\ell gm$, onde $m=|VG|+|aG|$ e d é o grau de entrada máximo de G .

TEOREMA 3 - Sejam G, E como definidos acima. Então

$$T(G,E) \leq d^{2^{m/c_3E} - 1}\ E\ c_7^{c_8^{m/E}} .$$

DEMONSTRAÇÃO - Por indução sobre $j = \lceil (m/E-1)/c_3 \rceil$.

Se $j \leq 0$ então $m \leq E$ e, portanto, o problema não envolve chamadas recursivas. Assim $(G,E) \leq E$ e o teorema segue trivialmente.

Se $j>0$ então $m>E$. Neste caso, o algoritmo chama a si próprio recursivamente com argumentos de tamanho (m',E'). Pelo Lema 2, $E' \geq c_5\ (\ell gd)\ m'/\ell gm'$ e do Lema 1 segue que $m'/E' \leq m/E - c_3$.

Assim,

$$\begin{aligned} j' &= \lceil (m'/E' - 1)/c_3 \rceil \\ &\leq \lceil (m/E - c_3 - 1)/c_3 \rceil \\ &= \lceil (m/E - 1)/c_3 \rceil - 1 \\ &= j - 1 \end{aligned}$$

Logo, o teorema pode ser aplicado indutivamente nas chamadas recursivas. Vamos considerar dois casos.

Caso 1: Corte Pequeno

São feitas duas chamadas recursivas, uma em G' e outra em G'', ambas com $\lceil (1-c_1)E \rceil$ pedras. Portanto,

$$T(G,E) \leq T(G', \lceil (1-c_1)E \rceil) + T(G'', \lceil (1-c_1)E \rceil)$$

$$\leq 2\, d^{2^{(m/E-c_3)/c_3}-1}\, E\, c_7\, c_8^{m/E-c_3}$$

$$\leq 2\, d^{2^{m/c_3E-1}-1}\, E\, c_7\, c_8^{m/E}$$

$$= (2/d)\, d^{2^{m/c_3E-1}}\, E\, c_7\, c_8^{m/E} .$$

Como $d \geq 2$ temos

$$T(G,E) \leq d^{2^{m/c_3E-1}}\, E\, c_7\, c_8^{m/E}$$

e, como $m/c_3E \geq 1$,

$$T(G,E) \leq d^{2^{m/c_3E}-1}\, E\, c_7\, c_8^{m/E} .$$

Caso 2: Corte Grande

Pelo Lema 2, temos $p \geq 1$. Assim,

$$p = \left\lfloor (2c_2 - c_3 - \frac{2(d+1)}{E} \frac{E^2}{dm} \right\rfloor - 1 \geq 2c_2 - c_3 - \frac{3d}{E} \frac{E^2}{3dm}$$

pois $\lfloor Z \rfloor - 1 \geq Z/3$, se $Z \geq 2$.

Da hipótese de que $E \geq c_5(\ell gd)m/\ell gm$ e de $m \geq m_o \geq 3$ segue que

$$\frac{3d}{E} \leq 3 \frac{d\ \ell gm}{c_5(\ell gd)m} \leq 3 \frac{d\ \ell gm_o}{c_5(\ell gd)(c_4 d)}$$

$$= 3 \frac{\ell gc_4 + \ell gd}{c_4 c_5\ \ell gd} \leq 3 \frac{(1+\ell gc_4)}{c_4 c_5}$$

e, utilizando (D10), temos

$$p \geq (2c_2 - c_3 - 3 \frac{1 + \ell gc_4}{c_4 c_5}) \frac{E^2}{3dm} \geq c_6 \frac{E^2}{dm} \qquad (1)$$

Da definição do algoritmo segue que

$$T(G,E) \leq T(G',\lfloor (E-dp)/2 \rfloor) + T(G',\lfloor (E-dp)/2 \rfloor \lceil T(G',\lfloor (E-dp)/2 \rfloor)/p \rceil$$

Aplicando a hipótese de indução, o Lema 1 e (1) obtemos

$$T(G,E) \leq d^{2^{m/c_3E-1}-1} E\, c_7^{c_8^{m/E-c_3}} +$$

$$d^{2^{m/c_3E-1}-1} E\, c_7^{c_8^{m/E-c_3}} \left\lceil \frac{dm}{c_6E^2} d^{2^{m/c_3E-1}-1} E\, c_7^{c_8^{m/E-c_3}} \right\rceil$$

Eliminando a função $\lceil\ \rceil$, aplicando a distributividade e reunindo termos semelhantes obtemos

$$T(G,E) \leq 2\, d^{2^{m/c_3E-1}-1}\, E\, c_7^{c_8^{m/E-c_3}} +$$

$$(1/c_6)\, E\, d^{2^{m/c_3E}-1}\, (m/E)\, c_7^{2c_8^{m/E-c_3}} .$$

De (D12) e de $m/E \geq 1$ segue que

$$T(G,E) \leq d^{2^{m/c_3E}-1}\, E(1+1/c_6)(m/E)\, c_7^{2c_8^{m/E-c_3}} \qquad (2)$$

Para obtermos a fórmula desejada devemos eliminar de (2) o fator $(1+1/c_6)(m/E)$ na linha da base, o fator 2 no expoente de c_7 e a subtração de c_3 no expoente de c_8. Para tanto, vamos mostrar que a adição de c_3 ao expoente de c_8 nos permite cancelar o fator $(1+1/c_6)(m/E)$ na linha da base e o fator 2 no expoente de c_7.

Vamos mostrar que se $m/E \geq 1$ então

$$c_7^{(c_8^{c_3}-2)}\, c_8^{m/E-c_3} \geq (1 + 1/c_6)(m/E) \qquad (3)$$

A derivada da função (de Z) $\frac{a^{b^Z}}{Z}$ é

$$\frac{a^{b^Z}}{Z^2}\,(\ell n a\ \ell n b\ Z\ b^Z - 1)$$

Se esta derivada for positiva para todo $Z \geq 1$, o mínimo da função para $Z \geq 1$ será seu valor no ponto $Z = 1$. Fazendo

$$a = c_7^{1-2/c_8^{c_3}}, \quad b = c_8 \quad \text{e} \quad Z = m/E \quad \text{obtemos}$$

$$\frac{a^{b^Z}}{Z} = (1 + 1/c_6)\,\frac{\text{lado esquerdo de (3)}}{\text{lado direito de (3)}} \tag{4}$$

As desigualdades (D12) e (D13) nos garantem que a derivada de (4) é positiva para $m/E \geq 1$. Assim, o mínimo da função é seu valor para $m/E = 1$. Da desigualdade (D11) segue que este valor é maior que $1 + 1/c_6$, o que prova (3) e o teorema.

□

COROLÁRIO 4 - Nas condições do Teorema 3

$$T(G,E) \leq E(c_7 d)^{c_8^{(d+1)n/E}}, \quad \text{onde } n = |VG|.$$

DEMONSTRAÇÃO - Seja $m = |aG| + |VG|$.

Do Teorema 3 e de (D13) segue que

$$\begin{aligned} T(G,E) &\leq d^{2^{m/c_3E}-1}\, E\, c_7^{c_8^{m/E}} \\ &\leq E\, d^{(2^{1/c_3})m/E}\, c_7^{c_8^{m/E}} \\ &\leq E\, d^{c_8^{m/E}}\, c_7^{c_8^{m/E}} \end{aligned}$$

$$\leq E(c_7 d)^{c_8^{m/E}}$$

$$\leq E(c_7 d)^{c_8^{(d+1)n/E}} .$$

□

3 - UM LIMITE INFERIOR

Nesta seção vamos construir uma família de grafos cuja ocupação em espaço $E \leq O(n)$ requer tempo

$$T \geq c_{10}\, E\, c_8^{c_9^{n/E}}$$

onde c_8, c_9, c_{10} são constantes positivas. Como na Seção 2.3, vamos utilizar a família $\{\bar{S}_N\}$ do Teorema 1.2.11.

Para estudar a ocupação daquela família utilizaremos constantes $c_1, c_2, \ldots, c_{10} > 0$ que satisfazem as seguintes desigualdades:

(D1) $c_1 < 1$

(D2) $9c_2 \leq c_1$

(D3) $c_3 \leq c_2/4$

(D4) $c_1 \leq 1 - 6c_3$

(D5) $c_4 \leq c_3/2$

(D6) $$c_5 < c_4/2$$

(D7) $$c_6 < 1$$

(D8) $$c_7 \geq 570 \left(\lceil 1/c_5 \rceil (1+c_6) + \frac{2c_6 \lceil 1/c_2 \rceil}{(1-c_6)\ln 2} \, 2^{\frac{1-c_6}{c_6}} - \frac{1}{\ln 2} \right)$$

(D9) $$c_8 \leq 2^{2-4c_5/c_4}$$

(D10) $$c_9 \leq 2^{c_6/c_7}$$

(D11) $$c_{10} \leq \frac{1-2c_3}{4} \lceil 1/c_5 \rceil$$

Um exemplo de constantes que satisfazem estas propriedades é

$c_1 = 9/11$, $c_2 = 1/11$, $c_3 = 1/44$, $c_4 = 1/88$, $c_5 = 1/180$,

$c_6 = 1/2$, $c_7 = 167211$, $c_8 = 1.03$, $c_9 = 1.000001$, $c_{10} = 42$.

Inicialmente, vamos estudar a família $\{G(N,k) : k \geq 1$ e N é múltiplo inteiro de $\lceil 1/c_2 \rceil \, 2^k\}$ construída indutivamente da seguinte maneira:

1. $G(N,1) = \ddot{S}_N$

2. para $k > 1$, $G(N,k)$ é composto de três cópias S^1, S^2 e S^3 de $\bar{S}_N$ e duas cópias G^1 e G^2 de $G(N/2, k-1)$, ligados como na Figura 3.1.

É imediato que $|VG(N,1)|=190N$ e para todo $k>1$, $|VG(N,k)| = 3(190N) + 2|VG(N/2,k-1)|$. Deixaremos a cargo do leitor a verificação de que $|VG(N,k)| = 190N(3k-2)$, para todo $k \geq 1$ e todo N múltiplo inteiro de $\lceil 1/c_2 \rceil 2^k$.

Se desdobrarmos a definição da família, veremos que $G(N,k)$ tem $2^{k+1}-3$ subgrafos dois a dois disjuntos que estão na família $\{\bar{S}_N\}_{N \geq 1}$. Assim, $G(N,k)$ pode ser representado por um grafo de barras, onde cada barra representa um superconcentrador (sem as arestas entre superconcentradores). A Figura 3.2 mostra um grafo de barras representando $G(N,4)$ com a respectiva numeração dada pela definição abaixo.

A cada superconcentrador S de $\{\bar{S}_N\}_{N \geq 1}$ em $G(N,k)$ vamos atribuir um *número de nível* $\nu_k(S)$ da seguinte maneira:

1. se $k=1$ então $S=G(n,1)$ e $\nu_k(S)=1$;
2. se $k>1$ então

   $\nu_k(S^1) = 1$;

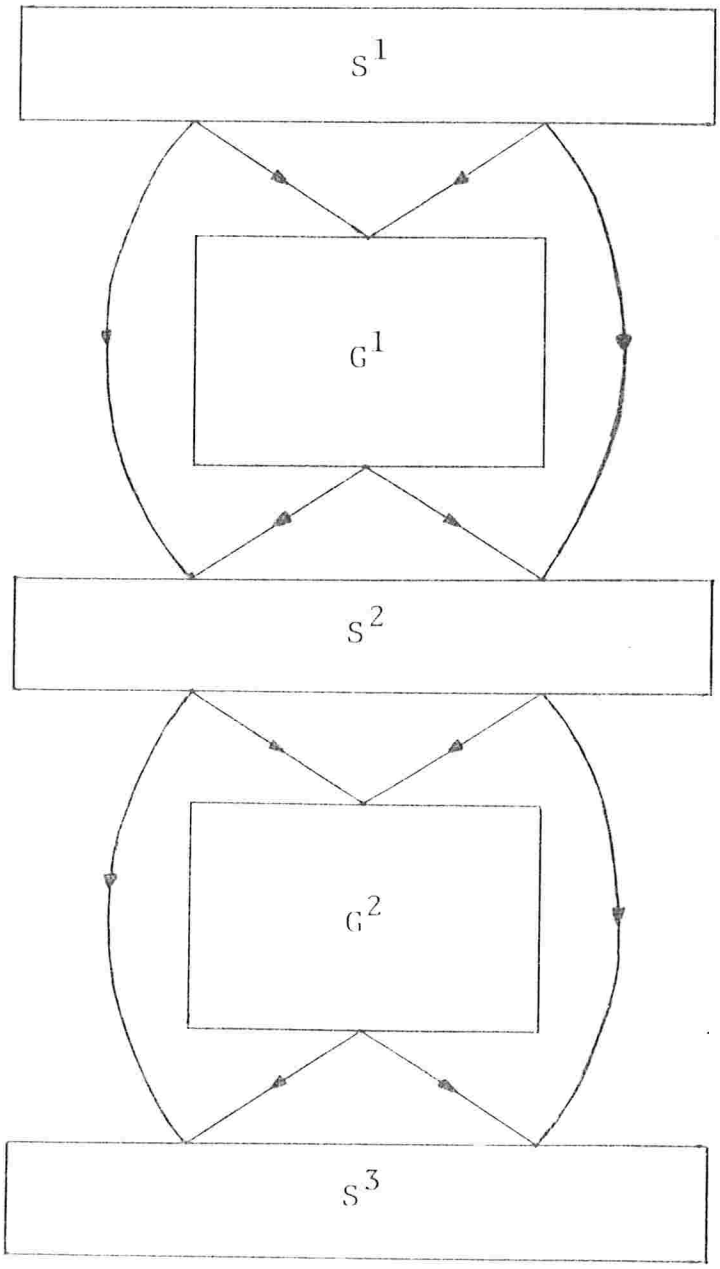
   para todo S em G^1, $\nu_k(S)=\nu_{k-1}(S)+1$;

   $\nu_k(S^2) = 2^k-1$;

   para todo S em G^2, $\nu_k(s)=\nu_{k-1}(S)+2^k-1$;

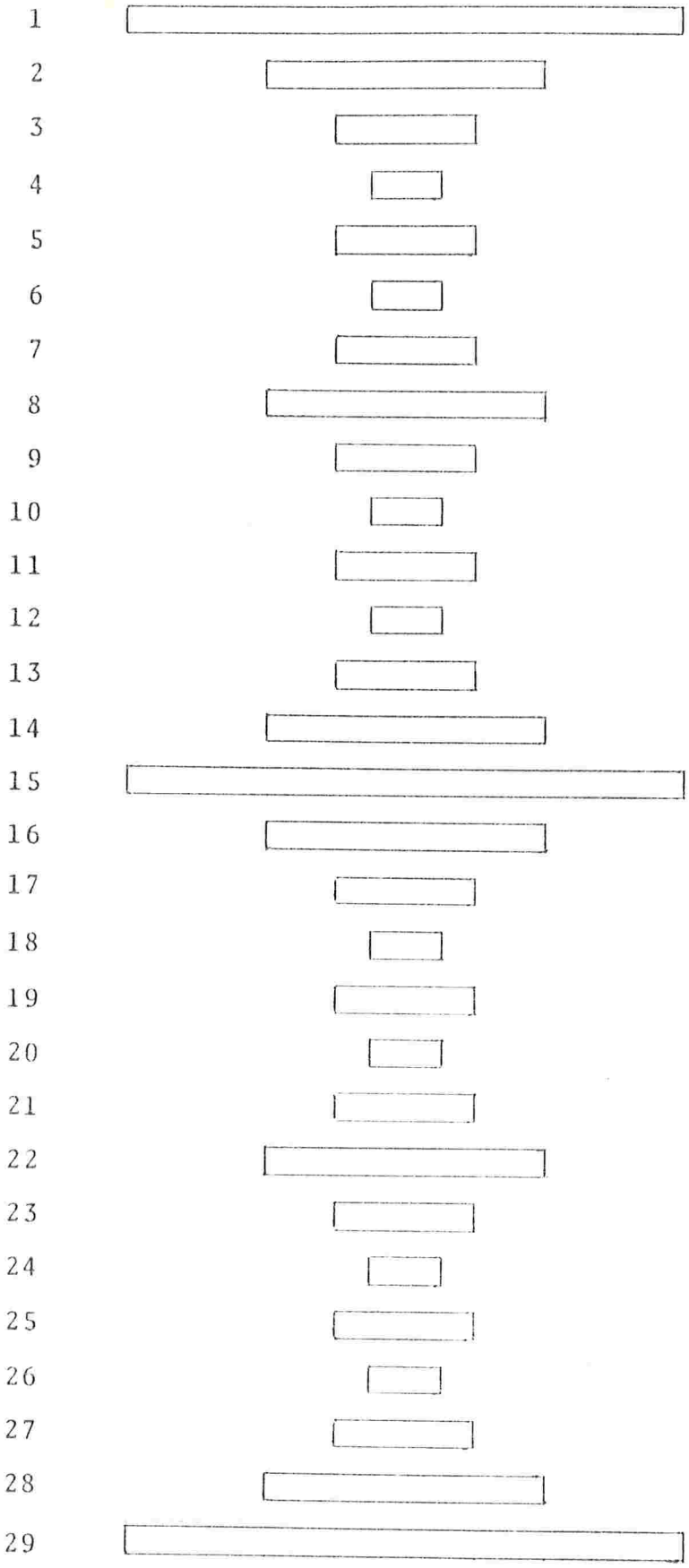
   $\nu_k(S^3) = 2^{k+1}-3$.

$G(N,k)$, $k > 1$



$\longrightarrow$ indica arestas ligando vértices correspondentes.

Figura 3.1



Representação esquemática de G(N,4) com números de nível.

Figura 3.2

Daqui para frente denotaremos o subgrafo de nível i por S_i. Diremos que S_i *precede* S_j ou S_j *sucede* S_i se $i < j$.

Vamos introduzir mais alguns conceitos sobre os superconcentradores S_i.

Seja $i > 1$ e consideremos o superconcentrador S_i.

O *pai* de S_i é o superconcentrador de maior nível que precede S_i e é maior que S_i, se existir; senão, o de mesmo tamanho. O fecho transitivo desta relação será chamada *relação de ancestralidade*.

Seja S_j o pai de S_i. A vizinhança de S_i é $\{S_\ell : j \leq \ell \leq i\}$

Note que o pai de S_i é maior que S_i a não ser que $i = 2^k - 1$ ou $i = 2^{k+1} - 3$.

Vamos assumir $E \leq c_5 N$. Como na Seção 2.3, vamos estudar a distribuição das pedras no grafo.

Consideremos um intervalo de tempo T.

Um N-superconcentrador S_i é *bom* em T se para todo sub-intervalo de T' de T durante o qual pelo menos $\lceil c_2 N \rceil$ sorvedouros de S_i são ocupados, o número de pedras na vizinhança de S_i é menor que $\lfloor c_3 N \rfloor$ em algum instante de T'.

Um N-superconcentrador S_i é *útil* em T se S_i e todos os seus ancestrais são bons em T e pelo menos $\lceil c_1 N \rceil$ sorvedouros de S_i são ocupados durante T.

Seja S' o pai do N'-superconcentrador S_i. Vamos supor que $S' \neq S_{i-1}$. Diremos que S' é *ótimo* para S_i em T se S' é útil em T e em cada instante de T há pelo menos $\lfloor c_4 N' \rfloor$ pedras na vizinhança de S_i.

Note que de $E \leq c_5 N$, (D1), (D5) e (D6) segue que $S_{2^{k+1}-3}$ é útil no intervalo $[1,T]$ de toda a ocupação de $G(N,k)$.

Mostramos a seguir que se S_i é útil em T, existem dois sub-intervalos disjuntos de T em cada um dos quais S_i é útil ou seu pai é ótimo para ele. Isto vai nos permitir iterar o Lema Fundamental de $S_{2^{k+1}-3}$ para S_1.

Note que para todo N múltiplo inteiro de $\lceil 1/c_2 \rceil 2, \lceil c_2 N \rceil \geq 2$. Além disso, por (D2)

$$\lceil c_1 N \rceil \geq 9c_2 N \geq 6\lceil c_2 N \rceil \qquad (0)$$

LEMA 1 - Consideremos um grafo $G(N,k)$ tal que $k>1$. Seja $i>1$. Vamos supor que durante um intervalo T, $\lceil c_2 N' \rceil$ sorvedouros do N'-superconcentrador S_i são ocupados, iniciando e terminando o processo com menos de $\lfloor 2c_3 N' \rfloor$ pedras na vizinhança de S_i. Vamos supor, ainda, que o pai de S_i seja um N''-superconcentrador. Então pelo menos $\lceil c_1 N'' \rceil$ sorvedouros do pai de S_i são ocupados durante T.

DEMONSTRAÇÃO - Pelo Lema Fundamental e por (D3), pelo menos $\lceil (1-4c_3)N' \rceil$ fontes de S_i são ocupadas e desocupadas durante T. Vamos considerar, então, três casos.

Caso 1: S_i é um S^1.

Neste caso, o pai de S_i é S_{i-1}, a vizinhança de S_i é $\{S_{i-1}, S_i\}$ e $N''=2N'$. Utilizando (D4) temos que, durante T pelo menos $\lceil (1-4c_3)2N' \rceil = \lfloor 2c_3 N' \rfloor \geq \lceil (1-5c_3)2N' \rceil \geq \lceil c_1 2N' \rceil = \lceil c_1 N'' \rceil$ sorve-

douros de S_{i-1} são ocupados.

Caso 2 - S_i é um S^2.

Neste caso, seja S' o S^1 correspondente, i.e., $S' = S_{i-2^k N''/N'-2}$. Devido às arestas de S' para S_i, pelo menos $\lceil(1-6c_3)N'\rceil \geq \lceil c_1 N\rceil$ sorvedouros de S' são ocupados durante T.

Se S' é o pai de S_i, o lema está provado. Caso contrário, o pai de S' é o pai de S_i e a vizinhança de S' está contida na vizinhança de S_i. Por (O), durante T pelo menos $\lceil c_1 N'\rceil \geq \lceil c_2 N'\rceil$ sorvedouros de S' são ocupados, iniciando e terminando o processo com menos de $\lfloor 2c_3 N'\rfloor$ pedras na vizinhança de S'. Aplicando o Caso 1 a S' obtemos o lema.

Caso 3 - S_i é um S_3.

Análogo ao Caso 2. □

LEMA 2 - Consideremos um grafo G(N,k) tal que k > 1. Seja i>1. Se o N'-superconcentrador S_i é útil num intervalo T então existem T_1, $T_2 \subseteq T$ disjuntos tais que o pai de S_i é útil em T_1 e em T_2.

DEMONSTRAÇÃO - Seja S' o N''-superconcentrador que é o pai de S_i. Todos os ancestrais de S_i são bons em T e, portanto, em todo sub-intervalo de T. Como os ancestrais de S' são os de S_i, resta-nos apenas provar que existem T_1, $T_2 \subseteq T = [t_1, t_2]$ disjuntos tais que pelo menos $\lceil c_1 N''\rceil$ sorvedouros de S' são ocupados durante cada um deles.

Por (O), $\lceil c_1 N'\rceil \geq 6\lceil c_2 N'\rceil$, i.é, pelo menos $6\lceil c_2 N'\rceil$ sorve

douros de S_i são ocupados durante T.

Sejam t_3, t_4, t_5 tais que durante cada um dos intervalos $[t_1,t_3]$, $[t_3+1,t_4]$ e $[t_4+1,t_5]$, exatamente $\lceil c_2N' \rceil$ sorvedouros de S_i são ocupados.

Como S_i é bom em T, existem $t_6 \varepsilon [t_1,t_3]$ e $t_7 \varepsilon [t_4+1, t_5]$ nos quais há menos de $\lfloor c_3N' \rfloor$ pedras na vizinhança de S_i. Durante $T_1=[t_6+1,t_7] \supseteq [t_3+1,t_4]$, pelo menos $\lceil c_2N' \rceil$ sorvedouros de S_i são ocupados, iniciando e terminando esse processo com menos de $\lfloor c_3N' \rfloor$ pedras na vizinhança de S_i. Pelo Lema 1, pelo menos $\lceil c_1N'' \rceil$ sorvedouros de S' são ocupados durante T_1.

Analogamente, existe $T_2 \subseteq [t_5+1,t_2]$ durante o qual pelo menos $\lceil c_1N'' \rceil$ sorvedouros de S' são ocupados.

□

LEMA 3 - Considere um grafo G(N,k) tal que k>1. Seja i>1. Seja S_i um N'-superconcentrador útil num intervalo T. Então existem T_1, $T_2 \subseteq T$ disjuntos tais que para j=1,2, S_{i-1} é útil em T_j ou o pai de S_i é ótimo para S_i em T_j.

DEMONSTRAÇÃO - Se S_{i-1} é o pai de S_i, o lema segue diretamente do Lema 2. Vamos supor, então, que S_{i-1} não é o pai de S_i,

Neste caso, $|VS_i| = 2|VS_{i-1}|$. Como S_i é útil em T, pelo menos $6\lceil c_2N' \rceil$ sorvedouros de S_i são ocupados durante T(por (0)). Seja T_1 o menor sub-intervalo de T em que os primeiros $3\lceil c_2 N' \rceil$ sorvedouros de S_i são ocupados e seja T_2 o menor sub-intervalo de T em que os últimos $3\lceil c_2N' \rceil$ sorvedouros de S_i são ocupados. Os argumentos abaixo podem ser aplicados a T_1 ou T_2 e vamos, sem perda de generalidade, considerar apenas T_1.

Seja t_1 o primeiro e t_2 o último instante de T_1 em que menos de $\lfloor c_3N' \rfloor$ pedras estão na vizinhança de S_i. Como S_i é bom, pelo menos $\lceil c_2N' \rceil$ sorvedouros de S_i são ocupados durante $[t_1+1, t_2]$. Vamos considerar dois casos.

<u>Caso 1</u> - S_{i-1} "não é bom" em $[t_1+1, t_2]$.

Existe $[t_3,t_4] \subseteq [t_1+1,t_2]$ durante o qual pelo menos $\lceil c_2N'/2 \rceil$ sorvedouros de S_{i-1} são ocupados e em cada instante do intervalo pelo menos $\lfloor c_4N' \rfloor$ pedras estão na vizinhança de S_{i-1}.

Seja $t_5 \varepsilon [t_1,t_3]$ o último instante em que existem menos de $\lfloor c_3N' \rfloor$ pedras na vizinhança de S_i. Seja $t_6 \varepsilon [t_4,t_2]$ o primeiro instante em que existem menos de $\lfloor c_3N' \rfloor$ pedras na vizinhança de S_i. Durante $[t_5+1, t_6]$ pelo menos $\lceil c_2(N'/2) \rceil$ sorvedouros de S_{i-1} são ocupados, iniciando e terminando o processo com menos de $\lfloor c_3N' \rfloor = \lfloor 2c_3(N'/2) \rfloor$ pedras na vizinhança de S_i (e, portanto, na de S_{i-1}). Pelo Lema 1, pelo menos $\lceil c_1N' \rceil$ sorvedouros do pai de S_{i-1} são ocupadas durante $[t_5+1, t_6]$. Por (0), $\lceil c_1N' \rceil \geq 6\lceil c_2N' \rceil$. Assim, podemos aplicar novamente o Lema 1 para obter que pelo menos $\lceil c_1(2N') \rceil$ sorvedouros do avô de S_{i-1}, i.e., pai de S_i são ocupados durante $[t_5+1, t_6]$. Por hipótese, em cada instante de $[t_5+1, t_6]$ há pelo menos $\lfloor c_4N' \rfloor$ pedras na vizinhança de S_i. Como os ancestrais de S_i incluem os de seu pai e S_i é útil em T_1, segue que o pai de S_i é ótimo para S_i em $[t_5+1, t_6]$.

<u>Caso 2</u> - S_{i-1} é "bom" em $[t_1+1, t_2]$.

Em qualquer sub-intervalo de $[t_1+1, t_2]$ durante o qual pelo menos $\lceil c_2N'/2 \rceil$ sorvedouros de S_{i-1} são ocupados existe

um instante em que há menos de $\lfloor c_4N' \rfloor \leq \lfloor c_3N'/2 \rfloor$ pedras na vizinhança de S_{i-1}.

Neste caso, S_{i-1} é bom em T_1. Vamos considerar dois casos.

Caso 2.1 - O pai de S_i tem o mesmo tamanho que S_i.

O pai de S_i é o pai de S_{i-1} e, portanto, todos os ancestrais de S_{i-1} são bons em T_1. Durante $[t_1 + 1, t_2]$

$\lceil c_2N' \rceil \geq \lceil 4c_3N' \rceil \geq 2\lfloor c_3N' \rfloor + 1$ (por (D3)) sorvedouros de S_i são ocupados, iniciando e terminando o processo com menos de $\lfloor c_3N' \rfloor$ pedras em S_i. Pelo Lema Fundamental, pelo menos $N'-2\lfloor c_3N' \rfloor \geq \lceil (1-2c_3)N' \rceil$ fontes de S_i são ocupadas durante $[t_1+1, t_2]$. Devido às arestas de S_{i-1} para S_i, pelo menos $\lceil (1-4c_3)N'/2 \rceil \geq \lceil (1-6c_3)N'/2 \rceil \geq \lceil c_1N'/2 \rceil$ (por (D4)) sorvedouros de S_{i-1} são ocupados durante $[t_1+1, t_2] \subseteq T_1$. Logo, S_{i-1} é útil em T_1.

Caso 2.2 - O pai de S_i é maior que S_i.

Novamente, vamos considerar dois casos.

Caso 2.2.1 - O pai de S_{i-1} "não é bom" em $[t_1+1, t_2]$.

Existe $[t_3, t_4] \subseteq [t_1+1, t_2]$ durante o qual $\lceil c_2N' \rceil$ sorvedouros do pai de S_{i-1} são ocupados e em cada instante do intervalo há pelo menos $\lfloor c_4N' \rfloor$ pedras na vizinhança de S_i.

A demonstração deste caso é análoga à do Caso 1.

Caso 2.2.2 - O pai de S_{i-1} é "bom" em $[t_1+1, t_2]$.

Em qualquer sub-intervalo de $[t_1+1, t_2]$ durante o qual $\lceil c_2N' \rceil$ sorvedouros do pai de S_{i-1} são ocupados, existe um instante em que há menos de $\lfloor c_4N' \rfloor$ pedras na vizinhança de S_i.

Neste caso, o pai de S_{i-1} é bom em T_1 e, portanto, S_{i-1} e seus ancestrais são bons em T_1. O resto da demonstração é análoga à do Caso 2.1.

□

Para estudarmos as relações entre o tempo e o espaço de uma estratégia qualquer para ocupação de G(N,k) vamos associar a ela uma árvore binária.

Seja T o tempo gasto por uma estratégia para ocupação de G(N,k) em espaço E. A esta estratégia vamos associar uma árvore binária R tal que cada um de seus vértices é um par (i,T) onde $1 \leq i \leq 2^{k+1}-3$ e $T \subseteq [1,T]$ e

1. a raiz de R é $(2^{k+1}-3, [1,T])$;
2. cada vértice (i,T) tal que $i>1$ tem dois filhos (i_j,T_j) $(j=1,2)$, onde T_j é definido no Lema 3 e $i_j = i-1$, se S_{i-1} é útil em T_j, ou o nível do pai de S_i, se este é ótimo para S_i em T_j;
3. cada vértice $(1,T)$ é uma folha.

As folhas de R são vértices $(1,T_i)$ onde os T_i são dois a dois disjuntos e S_1 é útil em cada um deles. Por (O), pelo menos $6\lceil c_2N\rceil \leq \lceil c_1N\rceil$ sorvedouros de S_1 são ocupados em cada um dos T_i. Aplicando o Lema Fundamental duas vezes em cada T_i (veja demonstração do Lema 2) temos que $2\lceil(1-2c_3)N\rceil$ fontes de S_1 são ocupadas durante T_i. Assim, se o número de folhas de R é pelo menos b, o número de ocupações de fontes de S_1 é pelo menos $2\lceil(1-2c_3)N\rceil b$ (que também é um limite inferior para T) .

Vamos determinar um limite inferior para b.

LEMA 4 - Um caminho qualquer da raiz de R até uma de suas folhas tem comprimento maior ou igual a

$$(1 - 2c_5/c_4)2^{k+1}-3$$

DEMONSTRAÇÃO - Consideremos um caminho arbitrário em R cujos vértices são (i_1,T_1), $(i_2,T_2),\ldots,(i_r,T_r)$, nessa ordem, e tal que $i_1 = 2^{k+1}-3$, $T_1 = [1,T]$ e (i_r,T_r) é uma folha,

$T_r \subseteq T_{r-1} \subseteq \ldots \subseteq T_1$ e $i_1 > i_2 > \ldots > i_r = 1$.

Seja $I = \{i_1,i_2,\ldots,i_r\}$ e seja $J = \{1,2,\ldots,2^{k+1}-3\} - I$. Para cada $j \in J$, existe $\ell (1 \leq \ell < r)$ tal que $i_\ell > j > i_{\ell+1}$. Além disso, $S_{i_{\ell+1}}$ é o pai de S_{i_ℓ} (caso contrário $i_{\ell+1} = i_\ell + 1$, o que é impossível). Portanto, $S_{i_{\ell+1}}$ é ótimo para S_{i_ℓ} em T_ℓ. Segue que se S_{i_ℓ} é um $(N/2^{k-p})$-superconcentrador, em cada instante de T_ℓ há pelo menos $\lfloor c_4N/2^{k-p} \rfloor$ pedras na sua vizinhança. Esta tem, no máximo, $2^{p+1}-2$ níveis. Todos estes níveis exceto i_ℓ e $i_{\ell+1}$ estão em J. Assim, a razão entre o número mínimo de pedras na vizinhança de S_{i_ℓ} durante T_ℓ e $i_{\ell+1}-i_\ell-1$ é pelo menos $\lfloor c_4N/2^{k+1} \rfloor$.

Como $T_r \subseteq T_\ell$, para todo $\ell (1 \leq \ell < r)$, o limite inferior acima vale para T_r. Note que algumas pedras que estão no grafo foram consideradas em duas vizinhanças diferentes (as que estão nos S_{i_ℓ} tais que $1 < \ell < r$). Portanto, cada nível de J tem, em média, pelo menos $c_4N/2^{k+2}$ pedras em cada instante de T_r. Logo, o número mínimo de pedras no grafo durante T_r é pelo menos

$$|J| \left\lfloor c_4 N/2^{k+2} \right\rfloor$$

Como, por hipótese, $E \leq c_5 N$ segue que

$$|J| \leq \frac{c_5}{c_4} \; 2^{k+2}$$

e, portanto,

$$r = 2^{k+1}-3 - |J| \geq (1-2c_5/c_4)2^{k+1}-3 \; .$$

□

LEMA 5 - O número de folhas de R é pelo menos

$$2^{(1-2c_5/c_4)2^{k+1}-3}$$

DEMONSTRAÇÃO - Segue do Lema 4 e de que cada vértice de R que não é folha tem dois filhos.

□

TEOREMA 6 - A ocupação de $G(N,k)$ $(k>1)$ com $E \leq c_5 N$ pedras requer tempo

$$T \geq \frac{(1-2c_3)}{4} \; N \; 2^{(1-2c_5/c_4)\; 2^{k+1}}$$

DEMONSTRAÇÃO - Se $k=1$, o limite inferior acima é menor que $|S_N| = 190\,N$. Se $k>1$, segue do Lema 5.

□

Sejam p, E naturais tais que $p/\ell gp \leq E \leq p$. Seja

$k = \lceil c_6 p/E \rceil$ e seja N o único múltiplo inteiro de $\lceil 1/c_2 \rceil 2^k$ entre $\lceil 1/c_5 \rceil E$ e $\lceil 1/c_5 \rceil E + \lceil 1/c_2 \rceil 2^k - 1$. Defina $G'(p,E) = G(N,k)$. O número de vértices de $G'(p,E)$ é, no máximo,

$$570Nk \leq 570\ (c_6 p/E+1)(\lceil 1/c_5 \rceil E + \lceil 1/c_2 \rceil 2^{c_6 p/E+1})$$

$$= 570\ (\lceil 1/c_5 \rceil c_6 p + c_6 \lceil 1/c_2 \rceil \frac{p}{E}\ 2^{c_6 p/E+1} + \lceil 1/c_5 \rceil E$$

$$+ \lceil 1/c_2 \rceil 2^{c_6 p/E+1})$$

$$\leq 570\ p(\lceil 1/c_5 \rceil (1+c_6) + 2c_6 \lceil 1/c_2 \rceil \ell g p\, p^{c_6-1} + 2\lceil 1/c_2 \rceil p^{c_6-1})$$

$$= 570\ p(\lceil 1/c_5 \rceil (1+c_6) + 2\lceil 1/c_2 \rceil p^{c_6-1} (1+c_6 \ell g\ p))$$

A função (de p) $2\lceil 1/c_2 \rceil\ p^{c_6-1}\ (1+c_6\ \ell g\ p)$ tem máximo

$\dfrac{2\ c_6 \lceil 1/c_2 \rceil}{(1-c_6)\ell n 2}\ 2^{\frac{1-c_6}{c_6} - \frac{1}{\ell n 2}}$. Portanto, $|VG'(p,E)| \leq c_7\ p$, onde

c_7 é uma constante que satisfaz (D8).

TEOREMA 7 - A ocupação de $G'(p,E)$ em espaço E requer tempo T tal que

$$T \geq c_{10}\, E\, c_8^{\,c_9 n/E}$$

onde $n = |VG'(p,E)|$.

DEMONSTRAÇÃO - Pelo Teorema 6,

$$T \geq \left(\frac{1-2c_3}{4}\right) \lceil 1/c_5 \rceil E\, 2^{(2-4c_5/c_4)\, 2^{c_6 n/c_7 E}}$$

O resto da demonstração segue de (D9), (D10) e (D11). □

CAPÍTULO 4

O JOGO DAS PEDRAS E COMPLEXIDADE DE ALGORITMOS

1 - INTRODUÇÃO

Estudaremos neste capítulo algumas aplicações do jogo das pedras à Complexidade de Algoritmos. A idéia básica das aplicações é construir grafos que representam computações e, estudando as estratégias de ocupação destes grafos, obter resultados a respeito de tempo e espaço como recursos computacionais.

Na Seção 2, estudamos o problema de reconhecer linguagens recursivas por máquinas de Turing (MT). A nomenclatura e a notação utilizadas são as de [Si79] com a ressalva de que, a menos que seja especificado em contrário, entenderemos por MT uma MT determinística, com fita de entrada e várias fitas de trabalho infinitas dos dois lados.

Como exemplo de aplicação à Complexidade Concreta, escolhemos uma aplicação à multiplicação de matrizes, que é apresentada na Seção 3. Apenas para introduzir as idéias, estudamos inicialmente a multiplicação de uma matriz por um vetor.

2 - O JOGO DAS PEDRAS E MÁQUINAS DE TURING

Um problema central em Complexidade de Algoritmos é o de relacionar o tempo e o espaço necessários para reconhecer uma

linguagem recursiva qualquer. Um resultado imediato nesta área é que, para toda função t de $\mathbb{N}$ em $\mathbb{N}$, TEMPO(t) $\subseteq$ ESPAÇO (t), já que o número de células utilizadas por uma MT em cada uma de suas fitas numa computação não pode exceder o número de passos de computação.

Paterson ([Pa72]) mostrou que se L é uma linguagem reconhecível em tempo $t = \Omega(n^2)$ por uma MT de uma única fita de entrada/trabalho, então L é reconhecível em espaço $t^{1/2}$. Para obter este resultado, Paterson utilizou seqüências de cruzamentos (vide [Si 79]) para simular a MT original .

Infelizmente, o conceito de seqüências de cruzamentos aparentemente não se generaliza para máquinas com várias fitas. Assim, não se conhece nenhum resultado semelhante ao de Paterson para máquinas com várias fitas. Em particular, não se sabia sequer se era possível melhorar o resultado trivial até que Hopcroft, Paul e Valiant ([HPV77]) mostraram que TEMPO (t) $\subseteq$ ESPAÇO (t/lgt), para toda função t de $\mathbb{N}$ em $\mathbb{N}$, utilizando o Jogo das Pedras. Este resultado é apresentado nesta seção.

Um *bloco* é um conjunto de células consecutivas de uma fita. O *extremo direito (esquerdo)* de um bloco é sua célula mais à direita (esquerda). O *k-bloco* i é o bloco de células de posições ik, ik+1, ..., (i+1)k-1.

Seja τ uma função de $\mathbb{N}$ em $\mathbb{N}$. Uma MT *respeita τ-blocos* se para toda entrada w, durante a computação de M sobre w a fronteira entre dois $\tau(|w|)$-blocos quaisquer de cada fita só é cruzada em instantes que são múltiplos inteiros de $\tau(|w|)$.

Uma MT com um τ-*relógio* é uma MT com duas fitas de entrada: uma delas contém uma palavra w sobre o alfabeto de entrada; a outra, que chamaremos de relógio, contém um bloco de $\tau(|w|)$ células marcadas (que pode ser vista como uma representação unária de $\tau(|w|)$). Inicialmente, a cabeça de leitura do relógio está sobre seu extremo esquerdo. Quando a computação se inicia, esta cabeça passa a se mover em direção ao outro extremo. Sempre que alcança um dos extremos do bloco (em instantes que são múltiplos inteiros de $\tau(|w|)$), a cabeça passa a se mover em direção ao outro extremo. Utilizando o relógio desta maneira, a MT determina os instantes da computação que são múltiplos inteiros de $\tau(|w|)$.

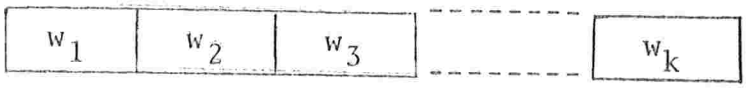
LEMA 1 - Sejam t, τ funções de $\mathbb{N}$ em $\mathbb{N}$ tais que para algum natural Z_o, $\tau(Z) \leq t(Z)$ para todo $Z \geq Z_o$. Seja M uma MT limitada em tempo por t. Então existe uma MT com τ-relógio M' que respeita τ-blocos e que simula M em tempo $O(t)$.

DEMONSTRAÇÃO - Vamos construir M'.

Para cada fita de trabalho de M, M' terá uma fita de trabalho com 4 trilhas. Além disso, M' terá uma fita de trabalho adicional, cuja utilidade será vista mais adiante.

Seja w uma entrada qualquer para M.

A idéia central da simulação (que será ligeiramente alterada posteriormente) é que se após $m\tau(|w|)$ passos da computação de M para w o conteúdo de sua j-ésima fita for



onde $|w_i| = \tau(|w|)$, o conteúdo da fita correspondente de M' após cm $\tau(|w|)$ passos será

1					
2		w_1^r	w_2^r		w_{k-1}^r
3	w_1	w_2	w_3		w_k
4	w_2^r	w_3^r	w_4^r		

onde x^r representa o reverso de x. Mais ainda, se a cabeça da fita de M estiver sobre sua p-ésima célula, a da fita de M' também estará.

Para que os extremos dos blocos possam ser detetados, colocaremos neles símbolos especiais, utilizando para isto a trilha 1 de cada fita.

A seguir, vamos descrever a simulação. Por simplicidade, vamos nos fixar numa fita F de M, já que o procedimento é igual para todas as fitas.

Inicialmente, M' marca os limites do primeiro bloco da fita F' de M' que corresponde a F, deixando sua cabeça na posição original (que é o extremo esquerdo do bloco).

Vamos supor que após m $\tau(|w|)$ passos de M, a cabeça de F esteja sobre sua p-ésima célula e que, após a simulação destes passos, a cabeça de F' também esteja (isto é verdade para m=0).

Lendo inicialmente a trilha 3, M' simula diretamente os próximos $\tau(|w|)$ passos de M. Durante esta simulação, ao in-

vés de cruzar as fronteiras do bloco, M' passa a ler a trilha correspondente no mesmo bloco.

Em seguida, de acordo com a idéia original, M' deveria atualizar os blocos vizinhos. Este processo é, em grande parte, desnecessário embora só aumente o tempo de simulação de um fator constante. Na verdade, só é necessário atualizar o próximo bloco a ser visitado e se este não foi o que acabou de ser visitado. Vamos supor, sem perda de generalidade, que o próximo bloco é o da direita.

A cabeça de F' marca a célula em que está (que será a próxima a ser lida) e se move até o extremo esquerdo do bloco. Em seguida, move-se até o extremo direito do bloco, copiando-o na fita adicional, e fica aguardando o próximo instante múltiplo inteiro de $\tau(|w|)$ para cruzar a fronteira. Percorre, então, o novo bloco atualizando-o e marcando seus limites. Finalmente, posiciona a cabeça na próxima célula a ser lida e fica aguardando o próximo instante múltiplo de $\tau(|w|)$ para simular os próximos $\tau(|w|)$ passos de M. Esta atualização custa $4\tau(|w|)$ passos.

Assim, cada intervalo de $\tau(|w|)$ passos de M é simulado por M' em, no máximo, $5\tau(|w|)$ passos. Portanto, a simulação toda (incluindo inicializações) pode ser feita em tempo $2\tau(|w|)+$ $+ 5t(|w|)$. □

Vamos descrever, agora, uma MT M_1 que será usada como procedimento por outra MT que construiremos mais adiante. A entrada de M_1 é uma codificação de um grafo G de $G(n,d)$ e sua saída é a sequência dos lances da estratégia MELHOR-OCUPAÇÃO

para a ocupação de G.

A MT M_1 tem uma fita de entrada/trabalho onde para cada vértice i de G ($1 \leq i \leq n$), há um bloco da forma

bin(i) * bin(i_1) * bin(i_2) * ... * bin(i_k)
e(i)
p(i)

onde i_1, i_2, ..., i_k são os vértices iniciais das arestas que chegam em i. Os blocos são armazenados segundo alguma ordenação topológica.

O campo e(i) indica se o vértice i está ou não ocupado. O campo p(i) é utilizado quando o vértice i pertence a algum conjunto V criado pelas chamadas recursivas do algoritmo.

A MT M_1 possui também duas fitas de trabalho que são utilizadas como pilhas e que chamaremos de P_1 e P_2.

A pilha P_1 será utilizada para a simulação das chamadas recursivas do algoritmo. Um contador k_1 é usado como apontador para o topo de P_1. Em cada elemento desta pilha é armazenado um endereço de retorno L_ℓ. Assim, "empilhar (G, V, L_ℓ) em P_1" significa escrever o primeiro e o último vértices de G e ℓ em P_1, somar 1 em k_1 e fazer $p(i) \leftarrow k_1$ para todo $i \in V$ tal que $p(i)=0$ (alguns destes vértices podem estar em outro conjunto correspondente a alguma chamada anterior do algoritmo). Além disso, um espaço $O(d\ell gn)$ é reservado para armazenar o conjunto U quando o grafo tiver no máximo m_o arestas. Indicaremos por

(G(j),V(j),L(j)) a tripla que está na posição j desta pilha.

A pilha P_2 armazena os subgrafos G' (i.e., seu primeiro e último vértices) cujos cortes correspondentes são grandes. Note que o correspondente V' não precisa ser guardado pois seus vértices estão no conjunto V da chamada que gerou o grafo G'.

Para ocupar ou desocupar um vértice i M_1 altera o campo e(i) e registra o lance na fita de saída. Além disso, faz p(i)←0 na desocupação.

A MT M_1 será utilizada na ocupação de grafos com a seguinte propriedade: para todo i ($1 \leq i < n$), i é predecessor de i+1 . Isto nos permitirá simplificar a construção de M_1 pois, quando o grafo for pequeno, i.e., tiver $m \leq m_0$ arestas, ele terá no máximo m+1 vértices e, portanto, pode ser ocupado com m+1 pedras, sem desocupações (veja demonstração do Teorema 2.2.3) . Além disso, quando o grafo é particionado e o corte gerado for grande, existe pelo menos uma aresta de G' para G", não sendo, portanto, necessária a ocupação de G' após a ocupação de G".

A MT M_1 procede de acordo com o seguinte algoritmo:

```
    início
    empilhe (G,∅,L1) em P1 ;
L2: m:=|aG(k1)|;
    se m ≤ m0 então
      início
L3:     se G(k1) está ocupado então
```

```
            início
            desocupe os vértices de VG(k1)-V(k1);                (1)
            vá para L(k1);
            fim
        senão
            início
            seja i o primeiro vértice (em ordem topológica) de
                                        VG(k1) não ocupado;
            U:={k:k é predecessor de i e não está ocupado};
L4:         se ∃ G'(j) em P2 tal que VG'(j)∩{u∈U:u não está ocu-
                                        pado} ≠ ∅ então
                início
                empilhe (G'(j),V'(j)∪U, L3) em P1;               (2)
                vá para L2;
L5:             desempilhe P1;
                vá para L4;
                fim
            senão
                início

                ocupe i;

                desocupe os vértices de U-VG(k1);                (3)
                vá para L3;
                fim
            fim
        fim
    senão
```

```
        início
        determine o último vértice de G';
        se |A| ≤ 2m/ℓgm então
           início
           W:={₁G(α):α∈A};
           empilhe (G',(V(k1)∩VG')∪W, L6) em P1;          (2)
           vá para L2;
L6:        desempilhe P1;
           empilhe (G'', V(k1)∩VG'', L7) em P1;           (4)
           vá para L2;
L7:        desempilhe P1;
           desocupe os vértices de W-(V(k1)∩VG');         (3)
           vá para L(k1);
           fim
        senão
           início
           empilhe G' em P2;
           empilhe (G'',V(k1)∩VG'', L8) em P1;            (4)
           vá para L2;
L8:        desempilhe P2;
           desempilhe P1;
           vá para L(k1);
           fim;
        fim;
L1: fim
```

□

(1) são os vértices i de $VG(k_1)$ tais que $p(i)=0$.

(2) o conjunto de vértices empilhado é U, pois os outros vértices i são tais que $p(i) \neq 0$.

(3) são os vértices i do grafo tais que $p(i)=k_1+1$.

(4) o conjunto de vértices empilhado é $\emptyset$, pois os outros vértices i são tais que $p(i) \neq 0$.

Para todo $m \geq 1$, seja $r(m)$ o maior nível de encaixamento das chamadas recursivas de MELHOR-OCUPAÇÃO para grafos com m arestas e grau de entrada máximo d.

É fácil ver que

$$r(m) = 1, \text{ se } m \leq m_o$$

$$r(m) \leq 1+\max\{\max\{r(m/2-m/\ell gm), r(m/2+m/\ell gm+d)\}, r(m/2-m/\ell gm)+r(m/2-m/\ell gm+d)\}, \text{ se } m>m_o$$

Da demonstração do Teorema 2.2.3 segue que $r(m) \leq O(m/\ell gm)$.

Assim, a pilha P_1 pode ter $O(n/\ell gn)$ elementos e seu armazenamento pode ser feito em espaço $O(n)$. Este também é um limite superior para o espaço necessário para armezenar P_2. Na fita de entrada/trabalho o espaço utilizado é $O(n\ \ell gn)$. Este espaço é suficiente para os cálculos de m, $|A|$ e $2m/\ell gm$.

Portanto, o espaço utilizado por M_1 é $O(n\ lgn)$.

LEMA 2 - Sejam t, τ funções de IN em IN tais que para algum $Z \geq Z_o$, $2 \leq lg^2(t(Z)) \leq \tau(Z) \leq t(Z)^{1/2}$ para todo $Z \geq Z_o$. Seja M' uma MT com τ-relógio que respeita τ-blocos e limitada em tempo por $t(Z)$. Então existe uma MT M'' com τ-relógio que reconhece T(M') em espaço $O(t/\ell gt)$.

DEMONSTRAÇÃO - Vamos construir M''.

A idéia básica da construção é associar a cada computação de M' um grafo e, utilizando a estratégia MELHOR-OCUPAÇÃO para ocupá-lo, simular M' em espaço $O(t/\ell gt)$.

Seja d o número de fitas de trabalho de M'. Para cada uma destas fitas, M' terá uma fita de trabalho.

Para descrever a simulação, vamos considerar uma entrada w para M' e vamos dividir a computação de M' sobre w em $n = \lceil t(|w|)/\tau(|w|)\rceil$ intervalos de $\tau(|w|)$ passos cada. A partir desta divisão construiremos o grafo.

A cada intervalo de tempo Δ vamos associar um vértice $v(\Delta)$. O grafo tem, portanto, n vértices. Para cada j $(1\leq j\leq d)$ e cada intervalo de tempo Δ, vamos denotar por Δ_j o último intervalo anterior a Δ em que a cabeça da j-ésima fita de M' visita o mesmo bloco que em Δ, se existir; caso contrário, vamos supor, por conveniência, que $\Delta_j=0$.

As arestas do grafo são tais que para todo $\Delta',\Delta(1\leq\Delta',\Delta\leq n)$, $v(\Delta')\rightarrow v(\Delta)$ sse $\Delta' = \Delta-1$ ou existe j tal que $\Delta'=\Delta_j$. É imediato que o grafo está em $G(n, d+1)$.

A cada vértice $v(\Delta)$ do grafo, vamos associar informações $h(\Delta)$, $b(\Delta)$ e $q(\Delta)$ definidas abaixo.

Seja $h(j,\Delta)$ a posição da cabeça da j-ésima fita de M' ao término do intervalo Δ. Seja $h(\Delta)=[h(1,\Delta),h(2,\Delta),\ldots,h(d,\Delta)]$ e seja $h=[h(1),h(2),\ldots,h(n)]$. A partir de h é possível construir o grafo, já que durante o intervalo $\Delta>1$ a cabeça da j-ésima fita visita o bloco $[h(j,\Delta-1)/\tau(|w|)]$.

Seja $b(j;\Delta)$ o conteúdo do bloco visitado pela cabeça da

j-ésima fita de M' durante o intervalo Δ, ao término deste. Seja $b(\Delta) = [b(1,\Delta), b(2,\Delta), \ldots, b(d,\Delta)]$.

Seja $q(\Delta)$ o estado de M' ao término do intervalo Δ e seja $q = [q(1), q(2), \ldots, q(n)]$. Para sabermos se M' aceita ou não w é suficiente determinar $q(n)$.

Como M' é determinística e respeita blocos, segue que $q(\Delta)$, $h(\Delta)$ e $b(\Delta)$ podem ser univocamente determinados a partir de $q(\Delta-1)$, $h(\Delta-1)$, $b(\Delta_1)$, $b(\Delta_2), \ldots, b(\Delta_d)$, pela simulação direta de $\tau(|w|)$ passos de M'.

Para que M'' construa o grafo da computação sobre w seria necessário que ela conhecesse h antes de iniciar a simulação, o que não é possível. Para contornar este problema, M'' gera em ordem lexicográfica, candidatos a h. Para cada candidato gerado, M'' o supõe correto e tenta simular a computação de M' sobre w. Se o candidato não é o correto, M'' deteta o erro e gera a próxima sequência; caso contrário, a computação de M' sobre w é simulada.

A MT M'' utiliza M_1 como uma co-rotina. A execução de M_1 ocorre por etapas, cada etapa correspondendo a um lance do jogo. Assim, cada vez que o controle é passado a M_1, ele é devolvido quando M_1 grava em sua fita de saída. Nesse instante, M'' guarda em seu controle finito o estado em que M_1 parou. Antes de chamar M_1 na próxima vez, M'' posiciona a cabeça da fita de saída de M_1 na sua célula mais à esquerda. A computação de M_1 prossegue então a partir de onde parou na chamada anterior. Note que M'' não altera o conteúdo das fitas de M_1.

Para verificar se um candidato h' é correto, M'' primeiramente constrói um grafo a partir de h' e procede da seguinte maneira:

```
início
enquanto v(n) não estiver ocupado faça
  início
  chame M1;
  se o próximo lance é "ocupe v(Δ)" então
     início
     compute e armazene h(Δ), b(Δ) e  q(Δ) ;
     se h'(Δ) ≠ h(Δ) então tente nova sequencia;
     fim
  senão     {próximo lance é"desocupe v(Δ)"}
     apague h(Δ), q(Δ) e b(Δ) das fitas
  fim
se q(n) é final então aceite w senão rejeite w
fim.
```

A existência das arestas de $v(\Delta-1)$ para $v(\Delta)$ garantem que o intervalo Δ não será simulado sem que os intervalos $1,2,\ldots,\Delta-1$ já tenham sido simulados alguma vez e, portanto, sem que já tinha sido verificado que $h'(1) = h(1)$, $h'(2) = h(2),\ldots$, $h'(\Delta-1) = h(\Delta-1)$.

Vamos mostrar que quando M_1 é chamada pela i-ésima vez, M'' tem computado e armazenado $\{b(\Delta), q(\Delta): v(\Delta)$ está ocupado no instante i da estratégia MELHOR-OCUPAÇÃO$\}$.

De fato, para $i=0$ a afirmação é verdadeira.

Para $i>0$, vamos considerar dois casos.

Se o $(i-1)$-ésimo passo da estratégia é "desocupe $v(\Delta)$" então antes da próxima chamada de M_1, $b(\Delta)$ e $q(\Delta)$ são apagados da fita e, da hipótese de indução, segue a tese.

Se o $(i-1)$-ésimo passo da estratégia é "ocupe $v(\Delta)$" então, no instante $i-1$ de MELHOR-OCUPAÇÃO, $v(\Delta_1)$, $v(\Delta_2)$,...,$v(\Delta_d)$ e $v(\Delta-1)$ estão ocupados. Por hipótese de indução, $b(\Delta_1)$,$b(\Delta_2)$, ..., $b(\Delta_d)$ e $q(\Delta-1)$ estão armazenados. Como $h(\Delta-1)$ também está armazenado, M'' pode simular os $\tau(|w|)$ passos de M' do intervalo para computar $b(\Delta)$ e $q(\Delta)$. Assim, também neste caso a tese é válida.

O armazenamento dos blocos pode ser feito em espaço $O(\tau(|w|)n/\ell gn)$, o da sequência h em espaço $O(n\ \ell gt)$ e o da sequencia q em espaço $O(n)$. Como já vimos antes, M_1 utiliza espaço de trabalho $O(n\ \ell gn)$. O espaço gasto em sua fita de saída é $O(\ell gn)$.

Para todo $Z\geq Z_o$, $2 \leq \lg(t(Z))^2 \leq \tau(Z) \leq t(Z)^{1/2}$ e, portanto,

$$n = \lceil t(Z)/\tau(Z) \rceil \leq 1+t(Z)/\tau(Z) \leq 2t(Z)/\tau(Z)$$

Assim, para todo $Z\geq Z_o$

$$c_1\ n\ \ell g(t(Z)) \leq 2\ c_1\ \frac{t(Z)}{\tau(Z)}\ \ell g(t(Z))$$

$$\leq 2\ c_1\ \frac{t(Z)}{\ell g^2(t(Z))}\ \ell g(t(Z))$$

$$\leq 2\ c_1\ \frac{t(Z)}{\ell g(t(Z))}$$

e

$$c_2\ \tau(Z)\ \frac{n}{\ell g n} \leq 2c_2\ \tau(Z)\ \frac{t(Z)/\tau(Z)}{\ell g(2t(Z)/\tau(Z))}$$

$$\leq 2c_2\ \frac{t(Z)}{\ell g(2t(Z)^{1/2})}$$

$$\leq 2c_2\ \frac{t(Z)}{\ell g(t(Z)^{1/2})}$$

$$\leq 4c_2\ \frac{t(Z)}{\ell g(t(Z))}$$

Logo, o espaço gasto por M'' é $O(t/\ell g\ t)$

□

Uma função t de $\mathbb{N}$ em $\mathbb{N}$ é *boa* se ela é construtível em tempo por uma MT M_t limitada em espaço por $O(t/\ell g\ t)$.

TEOREMA 3 - Seja t uma função boa e seja $\tau(Z) = \lceil t(Z)^{1/2} \rceil$. Se t e τ satisfazem as condições do lema anterior e M é uma MT limitada em tempo por t, então existe uma MT que simula M em espaço $O(t/\ell g\ t)$.

DEMONSTRAÇÃO - Pelos lemas 1 e 2, existe uma MT M'' com τ-relógio que simula M em espaço $O(t/\ell g\ t)$. Para obtermos o teorema, basta eliminar o relógio.

A Figura 4.1 mostra uma MT com uma única fita tal que para todo $Z \geq 0$, após Z passos M_2 tem marcado em sua fita um bloco de $\lceil Z^{1/2} \rceil$ células.

Ligando M_2 em paralelo com M_t obtemos uma MT que constrói um bloco de $\lceil t(Z)^{1/2} \rceil = \tau(Z)$ células na fita de M_2 em espaço $O(t/\ell g\ t)$.

A MT M_2

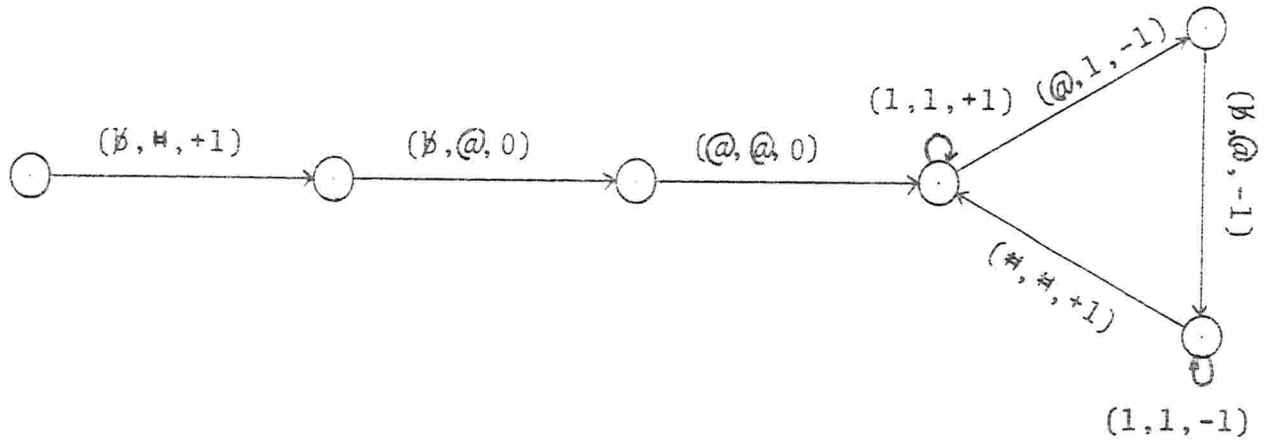


Figura 4.1

Ligando esta MT em série com M'' de tal maneira que a fita de M_2 seja o relógio de M'' obtemos uma MT que simula M em espaço $O(t/\ell g\ t)$.

□

COROLÁRIO 4 - Para toda função t de IN em IN, DTEMPO(t) $\subseteq$ DESPAÇO($t/\ell g\ t$).

DEMONSTRAÇÃO - Seja t uma função de IN em IN e seja M uma MT limitada em tempo por t.

Se t satisfaz as condições do teorema anterior, o corolário segue trivialmente. Caso contrário, podemos construir uma MT que faz toda a simulação do lema anterior para t(Z)=1,2,... até atingir um estado de aceitação ou rejeição e sem exceder o espaço $O(t/\ell g\ t)$.

□

Paul e Reischuk ([PR78]) obtiveram um resultado semelhante para máquinas de acesso aleatório.

3 - APLICAÇÕES À COMPLEXIDADE CONCRETA

Nesta seção apresentamos algumas aplicações do Jogo das Pedras visando obter limites inferiores para o tempo necessário para o cálculo de certos conjuntos de funções dependendo do espaço disponível. Estes resultados são, portanto, limites nas relações de troca entre tempo e espaço como recursos computacionais.

A técnica que apresentamos consiste em mostrar que grafos que representam certos conjuntos de funções contêm subgrafos que são (N,M)-superconcentradores e aplicar o Lema Funda-

mental a estes subgrafos.

O modelo de computação mais apropriado para estas aplicações é o de programas em linha reta. Seja D um conjunto qualquer. Seja F um conjunto de operações sobre D que contenha a função-identidade $I_D(d)=d$, para todo $d \in D$. Um *programa em linha reta sobre F* consiste de um conjunto de entradas $\{x_1, x_2, \ldots, x_N\}$ e uma sequência finita de atribuições a registradores, cada uma da forma $R_i \leftarrow x_j$ ou $R_i \leftarrow f(R_{j_1}, R_{j_2}, \ldots, R_{j_k})$ $(k \geq 0)$, onde $f \in F$ e $R_{j_1}, R_{j_2}, \ldots, R_{j_k}$ aparecem do lado esquerdo de alguma atribuição anterior. Além disso, cada entrada aparece pelo menos uma vez do lado direito de alguma atribuição.

Um programa em linha reta é uma codificação de um programa cuja execução consiste em, uma vez atribuídos valores às entradas, executar as atribuições na ordem em que elas aparecem. Executar uma atribuição significa armazenar no registrador o valor da expressão à direita.

Programas em linha reta são programas sem instruções de desvio. Uma das razões pelas quais este modelo é adotado é que em muitos programas as instruções de desvio servem apenas para fazer com que certos trechos do programa sejam repetidos um certo número de vezes. Duplicando estes trechos quantas vezes forem necessárias obtemos programas em linha reta equivalentes aos originais.

O número de instruções de um programa em linha reta é o *tempo* do programa e o número de registradores distintos que nele aparecem é o *espaço* do programa.

O programa *computa* um conjunto F de funções se, para to-

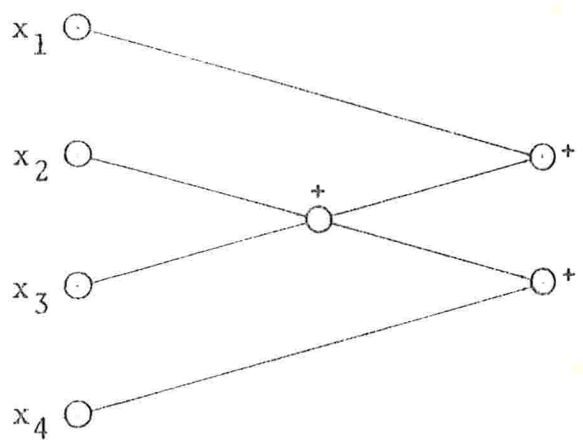
do $g \in F$, existe uma atribuição a um registrador R_i tal que para toda entrada $(x_1, x_2, \ldots, x_N) \in D^N$, após a execução dessa atribuição, R_i armazenará o valor de $g(x_1, x_2, \ldots, x_N)$. Note que este valor não precisa necessariamente ficar armazenado em R_i até o final da execução do programa. Isto corresponde a imaginar que o valor de g, uma vez armazenado em R_i, poderia ser impresso. As instruções de impressão não fazem parte do modelo pois, do ponto de vista da Complexidade, elas no máximo dobrariam o tempo do programa e, portanto, não alterariam sua ordem de grandeza.

Suporemos, sem perda de generalidade, que programas em linha reta sobre um conjunto F não tem atribuições da forma $R_i \leftarrow f(\ldots, R_i, \ldots)$. Atribuições deste tipo podem ser eliminadas de um programa com o auxílio de um novo registrador R_j e a substituição de cada atribuição $R_i \leftarrow f(\ldots, R_i, \ldots)$ pelas atribuições $R_j \leftarrow f(\ldots, R_i, \ldots)$ e $R_i \leftarrow R_j$, nesta ordem. O espaço do novo programa é o do programa original mais 1 e o tempo é no máximo o dobro, não sendo, portanto, alteradas as ordens de grandeza de ambos.

Podemos associar um grafo acíclico a um programa em linha reta da seguinte maneira: a cada entrada x_j corresponderá uma fonte de grafo; a cada instrução do tipo $R_i \leftarrow f(R_{j_1}, R_{j_2}, \ldots, R_{j_k})$, corresponderá um vértice no qual para todo $p (1 \leq p \leq k)$, chega uma aresta que sai do vértice correspondente a x_j se a última atribuição a R_{j_p} é $R_{j_p} \leftarrow x_j$ ou do vértice correspondente à última atribuição a R_{j_p}, em caso contrário. Este grafo será chamado de *grafo associado* ao programa. Abaixo, temos um exem-

plo de um programa e o grafo associado a ele.

$R_1 \leftarrow x_2$
$R_2 \leftarrow x_3$
$R_3 \leftarrow R_1 + R_2$
$R_2 \leftarrow x_1$
$R_1 \leftarrow R_3 + R_2$
$R_2 \leftarrow x_4$
$R_1 \leftarrow R_3 + R_2$



Note que vários programas em linha reta são associados a um mesmo grafo. Além disso, um programa em linha reta induz uma estratégia para ocupação do grafo associado a ele, se associarmos uma pedra a cada registrador. Reciprocamente, um grafo acíclico que representa uma computação de um conjunto F de funções usando operações de $\mathcal{F}$, e uma estratégia para sua ocupação induzem um programa em linha reta sobre $\mathcal{F}$ que computa F. O tempo do programa é o número de ocupações da estratégia e o espaço é o espaço da estratégia. Note que o tempo do programa está entre T'/2 e T', onde T' é o tempo da estratégia.

Assim, a ordem de grandeza do tempo T necessário para se computar um conjunto F de funções de D^N em D em espaço E é a mesma do tempo necessário para se ocupar algum grafo que representa um programa que computa F com E pedras.

Nesta seção, vamos determinar limites inferiores para o produto ET, fixado um conjunto F.

Vamos considerar um programa que computa F e seu grafo associado. Podemos supor, sem perda de generalidade, que o pro

grama não tem atribuições que não são usadas para a computação de F. Cada sorvedouro do grafo corresponde a uma atribuição do valor de alguma função de F nas entradas. No entanto, o cálculo de alguma função f de F pode ser utilizado para o cálculo de outra função de F, e o vértice correspondente à atribuição do valor de f a um registrador R_i não será um sorvedouro. Imediatamente após esta instrução, vamos acrescentar uma atribuição $R_j \leftarrow R_i$, onde R_j é um registrador que não aparece no programa original. O grafo associado ao novo programa assim obtido é o associado ao programa original acrescido de um sorvedouro e de uma aresta. Repetindo esta operação tantas vezes quantas necessárias, obtemos um programa que computa F em espaço E+1 e tempo no máximo $T+|F| \leq 2T$. Ambos tem, respectivamente, a mesma ordem de grandeza do espaço e do tempo do programa original.

Seja K o corpo dos números reais ou o dos números complexos.

Seja A uma matriz M×N sobre K. Para cada $I=\{i_1<i_2<\ldots<i_k\} \subseteq \{1,2,\ldots,M\}$ $(k \geq 1)$ e cada $J = \{j_1<j_2<\ldots<j_\ell\} \subseteq \{1,2,\ldots,N\} (\ell \geq 1)$, vamos denotar por A_{IJ} a matriz

$$\begin{bmatrix} a_{i_1 j_1} & a_{i_1 j_2} & \cdots & a_{i_1 j_\ell} \\ a_{i_2 j_1} & a_{i_2 j_2} & \cdots & a_{i_2 j_\ell} \\ \vdots & \vdots & & \vdots \\ a_{i_k j_1} & a_{i_k j_2} & \cdots & a_{i_k j_\ell} \end{bmatrix}$$

Uma tal matriz será chamada *submatriz* de A.

Diremos que A é *admissível* se todas as suas submatrizes

quadradas são não-singulares.

Seja F um conjunto de operações sobre K. Se toda função de F é linear, um programa em linha reta sobre F será chamado *programa linear*. O grafo associado a ele será chamado *grafo linear*.

Vamos estudar, agora, o produto $y=Ax$, onde $x \in K^N$. Supondo, sem perda de generalidade, que as linhas de A são 2 a 2 distintas, podemos considerar o cálculo do produto $y=Ax$ como o de M funções lineares nas variáveis $x_1, x_2, \ldots, x_N$. Um grafo linear que representa este produto tem pelo menos N fontes, que correspondem às entradas e às atribuições do tipo $R_i \leftarrow 0$ (a função nula é a única função constante que é linear). Além disso, podemos supor, sem perda de generalidade, que tal grafo tem M sorvedouros.

O teorema abaixo nos será de grande utilidade para mostrar que certos grafos são superconcentradores.

TEOREMA 1 (Teorema de Menger) - Seja G um grafo acíclico. Sejam V', V'' subconjuntos disjuntos e não-vazios de VG. Então $\max\{k : \exists$ um k-fluxo de V' para $V''\} = \min\{\ell : \exists\ \ell$ vértices em G tais que todo caminho de V' para V'' passa por um deles$\}$.

DEMONSTRAÇÃO - Vide [Be 79], pg. 167.

□

TEOREMA 2 - Seja G um grafo linear que representa o produto $y = Ax$. Sejam $I \subseteq \{1,2,\ldots,M\}$ e $J \subseteq \{1,2,\ldots,N\}$ tais que $|I| = |J|$. Sejam X o conjunto das fontes correspondentes aos $x_j (j \in J)$ e Y o conjunto dos sorvedouros correspondentes aos $y_i (i \in I)$. Se A_{IJ} é não-singular então existe um $|X|$-fluxo de X

para Y.

DEMONSTRAÇÃO - Vamos supor, sem perda de generalidade, que os $x_j (j \notin J)$ são nulos. Neste caso, podemos considerar os $y_i (i \in I)$ como funções lineares de $K^{|J|}$ em K^1.

A matriz A_{IJ} é a matriz de representação dos y_i em relação à base canônica do espaço das funções lineares de $K^{|J|}$ em K^1. Como A_{IJ} é não-singular, segue que os y_i são linearmente independentes.

Se existirem $|X|-1$ vértices em G cuja remoção desconecta Y de X, cada um dos $y_i (i \in I)$ pode ser escrito como combinação linear das funções lineares cujo valor fica armazenado após a execução das atribuições correspondentes a esses $|X|-1$ vértices. Isto contradiz a independência linear dos y_i.

Assim, pelo menos $|X|$ vértices devem ser removidos de G para desconectar Y de X. Pelo Teorema 1, existe um $|X|$-fluxo de X para Y.

□

COROLÁRIO 3 - Seja G um grafo linear que representa o produto $y = Ax$. Se A é admissível então G tem um subgrafo que é um (N,M)-superconcentrador cujas fontes são as correspondentes às entradas e os sorvedouros são os de G.

DEMONSTRAÇÃO - Segue do Teorema 2.

□

COROLÁRIO 4 - Se A é admissível então o cálculo do produto $y = Ax$ por um programa linear em espaço E requer tempo

$$T \geq E + \frac{M(N-E)}{E+1}$$

DEMONSTRAÇÃO - Segue do Corolário 3 e do Lema 1.2.4.

□

Vamos, agora, generalizar estas idéias para o produto de matrizes C = AB, onde A e B são, respectivamente, matrizes M× ×N e N×P sobre K.

Supondo, sem perda de generalidade, que as linhas de A são duas a duas distintas, podemos considerar o cálculo deste produto como o de MP funções lineares nas variáveis b_{ij}. Um grafo linear que representa este produto tem pelo menos NP fontes . Além disso, podemos, sem perda de generalidade, supor que tal grafo tem MP sorvedouros.

Seja G um grafo linear que representa o produto C=AB. Para todo i ($1 \leq i \leq P$), seja X_i o conjunto das fontes de G correspondentes aos elementos da i-ésima coluna de B e seja Y_i o conjunto dos sorvedouros de G correspondentes aos elementos da i-ésima coluna de C.

TEOREMA 5 - Se A é admissível então para todo i ($1 \leq i \leq P$), existe um subgrafo S_i de G que é um (N,M)-superconcentrador cujo conjunto de fontes é X_i e cujo conjunto de sorvedouros é Y_i.

DEMONSTRAÇÃO - Seja i tal que $1 \leq i \leq P$.

A i-ésima coluna de C é o produto (de matriz por vetor)de A pela i-ésima coluna de B. Assim, fazendo todas as entradas de B nulas exceto as da i-ésima coluna e eliminando os vértices dos quais não sai nenhum caminho que chega em Y_i, obtemos um subgrafo de G que é um grafo associado a um programa que computa o produto de uma matriz por um vetor.

Aplicando o Corolário 3 a este subgrafo obtemos a tese.

□

COROLÁRIO 6 - Se A é admissível então a ocupação de G em espaço E ($0 < E < N,M$) requer tempo T tal que

$$T \geq PE + \frac{PM(N-E)}{E+1}$$

DEMONSTRAÇÃO - Seja i tal que $1 \leq i \leq P$.

Uma estratégia para ocupação de G em espaço E induz uma estratégia para ocupação de S_i. Pela demonstração do Lema 1.2.4, a ocupação de G requer pelo menos $E + \frac{M(N-E)}{E+1}$ ocupações de fontes de S_i.

Como os conjuntos das fontes dos S_i são dois a dois disjuntos, temos que a ocupação de G requer tempo

$$T \geq PE + \frac{PM(N-E)}{E+1}$$

□

CAPÍTULO 5

CONCLUSÃO

A demonstração da existência de (N,6,4,6)-concentradores e a construção recursiva de N-superconcentradores são de Pippenger ([Pi77]). Pippenger também mostrou que seus N-superconcentradores tem no máximo 40N vértices. A contagem do número de vértices que apresentamos aqui, além de mais genérica, nos leva a um limite superior de 22N para o número de vértices do N-superconcentrador de Pippenger. O Lema 1.2.1 é de Paul,Tarjan e Celoni ([PTC77]). O Lema Fundamental e o Lema 1.2.4 são de Tompa ([To78]). A versão alternativa do Lema Fundamental é de Lengauer ([Le79]).

Hopcroft, Paul e Valiant ([HPV77]) mostraram que $E_d(n) = O(n/\ell gn)$. De sua demonstração, Paul,Tarjan e Celoni derivaram a primeira versão do algoritmo MELHOR-OCUPAÇÃO. A versão de MELHOR-OCUPAÇÃO apresentada aqui é do autor, bem como a demonstração de sua correção. A análise do algoritmo é essencialmente a que aparece em [PTC77]. Paul, Tarjan e Celoni mostraram, ainda, que $E_d(n) = \Omega(n/\ell gn)$.

As relações entre tempo e espaço no Jogo das Pedras são de Lengauer.

O resultado do Corolário 4.2.4 é de Hopcroft, Paul e Valiant. Em sua demonstração eles utilizaram uma MT não-determi

nística para gerar os lances da ocupação do grafo e as técnicas de Savitch ([Sa70]) para torná-la determinística. A construção da MT M_1 é do autor. Além disso, introduzimos o modelo de MT com relógio com a intenção de tornar mais claras as demonstrações dos Lemas 4.2.1 e 4.2.2.

As relações entre programas em linha reta e grafos acíclicos e o Teorema 4.3.2 são de Valiant ([Va76]). O Corolário 4.3.6 aparece em [Ja80] . No entanto, nos parece que sua demonstração exige uma hipótese a mais, o que o torna mais fraco. A demonstração que apresentamos é do autor.

Uma variação do jogo foi introduzida por Cook e Sethi ([CS76]). As pedras podem ser brancas ou pretas; as pedras brancas representam passos não-determinísticos de algoritmos . Outra variação é devida a Emde-Boas e Leeuwen ([EL79]). Nela uma terceira regra é acrescida às do jogo original: se todos os predecessores de um vértice v estão ocupados então uma pedra pode ser movida de um predecessor de v para ele. Emde-Boas e Leeuwen provaram que um grafo pode ser ocupado em espaço E segundo as novas regras sse ele pode ser ocupado em espaço E+1 segundo as regras originais.

Paul e Reischuk ([PR79]) obtiveram um resultado semelhante ao Corolário 4.2.4 para a simulação de uma máquina de acesso aleatório por uma MT.

Valiant aplicou os resultados sobre o Jogo das Pedras à Transformada Discreta de Fourier e Ja' Ja' à Multiplicação de Matrizes e à Inversão de Matrizes. Outras relações do Jogo das Pedras com Complexidade Concreta aparecem em [To78] e [CS 79].

Outros aspectos da complexidade do jogo são encontradas em [Se75] , [Li78] e [GLT79].

Alguns problemas relacionados com o Jogo das Pedras permanecem, ainda, em aberto:

1. construção explícita de (N,d,c_1,c_2)-concentradores;
2. determinação das constantes ótimas dos Capítulos 2 e 3;
3. TEMPO $\stackrel{?}{=}$ ESPAÇO$(t/\ell gt)$;
4. aplicação do Jogo das Pedras a outros problemas de Complexidade Concreta (p. ex., Cálculo do Determinante).

REFERÊNCIAS

[Be79] C. Berge
Graphs and Hypergraphs
North Holland (1979)

[Co73] S.A. Cook
An observation of time-space tradeoff
5th ACM-STOC (1973), 29-33

[CS76] S.A.Cook, R. Sethi
Storage requirements for deterministic polynomial finite recognizable languages
JCSS 13 (1976), 25-37

[CSa80] D.A. Carlson, J.E. Savage
Graph pebbling with many free pebbles can be difficult
12th ACM-STOC (1980), 326-332

[EL79] P. van Emde-Boas, J. van Leeuwen
Move rules and time-space tradeoffs in the pebble game
Lecture Notes in Computer Science 67 (1979), 101-112

[GG79] O. Gabber, Z. Galil
Explicit construction of linear size concentrators and superconcentratrators
20th IEEE-FOCS (1979), 364-370

[GLT79] J.R. Gilbert, T.Lengauer, R.E. Tarjan
The pebbling problem is complete in polynomial spce
1. 11th ACM-STOC (1979), 237-248
2. SIAM JC 9 (1980), 513-524

[HP70] C.E. Hewitt, M.S. Paterson
Comparative schematology
Conf. on Concurrent Systems and Parallel Computation (1970), 119-127

[HPV77] J.E. Hopcroft, W.J. Paul, L.G. Valiant
On time versus space
JACM 2 (1977), 332-337

[Ja80] Joseph Ja' Ja'
Time-space tradeoffs for some algebraic problems
12th ACM-STOC (1980), 339-350

[Le79] T. Lengauer
Upper and lower bounds for time-space tradeoffs in a pebble game
Ph. D. Thesis, Stanford University (1979)

[Li78] A. Lingas
A P-space problem related to a pebble game
Lecture Notes in Computer Science 62 (1978),300-321

[LT79] T. Lengauer, R.E. Tarjan
Upper and lower bounds on time-space tradeoffs
11th ACM-STOC (1979), 262-277

[Lu79] C.L. Lucchesi
Introdução à Teoria dos Grafos
12º Colóquio Brasileiro de Matemática - IMPA (1979)

[Pa72] M.S. Paterson
Tape bounds for time-bounded Turing Machines
JCSS 6 (1972), 116-124

[Pi77] N. Pippenger
Superconcentrators
SIAM JC 6 (1977), 298-304

[Pi78] N. Pippenger
A time-space tradeoff
JACM 25 (1978), 509-515

[PR79] W. J. Paul, R. Reischuk
On time versus space II
20th IEEE-FOCS (1979), 298-306

[PT77] W.J. Paul, R.E. Tarjan
Time-space tradeoffs in a pebble game
Lecture Notes in Computer Science 52 (1977), 365-369

[PTC77] W.J. Paul, R.E. Tarjan, J.R. Celoni
Space bounds for a game on graphs
Math. Systems Theory 10 (1977), 239-251

[PV76] N. Pippenger, L.G. Valiant
Shifting graphs and their applications
JACM 23 (1976), 423-432

[Re78] R. Reischuk
Improved bounds on the problem of time-space tradeoffs in a pebble game
1. 19th IEEE-FOCS (1978), 83-93
2. JACM 27 (1980), 839-849

[Sa70] W.J. Savitch
Relationships between nondeterministic and deterministc tape complexity
JCSS 4 (1970), 177-192

[Se75] R. Sethi
Complete register allocation problems
SIAM JC 4 (1975), 226-248

[Si79] I. Simon
Introdução à Teoria de Complexidade de Algoritmos
I Escola de Computação (1979), São Paulo

[To78] M. Tompa
Time-space tradeoffs for computing functions using connectivity properties of their circuits
1. 10th ACM-STOC (1978), 196-204
2. JCSS 20 (1980), 118-132

[Va76] L.G. Valiant
Graph-theoretic properties in computational complexity
JCSS 13 (1976), 278-285