TIME-SPACE TRADE-OFFS IN A PEBBLE GAME

W.J. Paul[1)]
Fakultät für Mathematik
der Universität Bielefeld
D-48oo Bielefeld 1
Germany

R.E. Tarjan[2)]
Computer Science Department
Stanford University
Stanford, Ca. 943o5
USA

Abstract: A certain pebble game on graphs has been studied in various contexts as a model for time and space requirements of computations [1,2,3,7]. In this note it is shown that there exists a family of directed acyclic graphs G_n and constants c_1, c_2, c_3 such that

1) G_n has n nodes and each node in G_n has indegree at most 2 .
2) Each graph G_n can be pebbled with $c_1\sqrt{n}$ pebbles in n moves.
3) Each graph G_n can also be pebbled with $c_2\sqrt{n}$ pebbles, $c_2 < c_1$, but every strategy which achieves this has at least $2^{c_3\sqrt{n}}$ moves.

Let $S(k,n)$ be the set of all directed acyclic graphs with n nodes where each node has indegree at most k . On graphs $G \in S(n,k)$ the following one person game is considered. The game is played by putting pebbles on the nodes of G according to the following rules:

i) an input node (i.e. a node without ancestor) can always be pebbled.
ii) if all immediate ancestors of a node c have pebbles one can put a pebble on c .
iii) one can always remove a pebble from a node.

Goal of the game is to put according to the rules a pebble on some output node (i.e. a node without successor) of G in such a way, that the total number of pebbles which are simultaneously on the graph is minimized.

The game models time and space requirements of computations in the following sense. The nodes of G correspond to operations and the pebbles correspond to storage locations. If a pebble is on a node this means that the result of the operation to which the node corresponds is stored in some storage location. Thus the rules have the following meaning:

---

1) Research partially supperted by DAAD (German Academic Exchange Service) Grant No. 43o/4o2/653/5.
2) Research partially supported by the National Science Foundation, Grant No. MCS 75-2287o and by the Office of Naval Research, Contract No. N oo14-76-C-o688.

i) input data are always accessible.

ii) if all operands of an operation are known and stored somewhere the operation can be carried out and the result be stored in a new location.

iii) storage locations can always be freed. By the rules a single node can be pebbled many times. This corresponds to recomputation of intermediate results.

In particular the game has been used to model time and space of Turing machines [1,2] as well as length and storage requirements for straight line programs [7].

Known results about the pebble game include

A: Every graph $G \in S(k,n)$ can be pebbled with $c_k n/\log n$ pebbles where the constant c_k depends only on k [2].

B: There is a constant c and a family of graphs $G_n \in S(2,n)$ such that for infinitely many n G_n cannot be pebbled with less than $cn/\log n$ pebbles [4].

For more results see [1,3,4,7].

By putting pebbles on the nodes of a graph G in topological order (i.e. if there is an edge from node c to node c' then c is pebbled first) one can pebble each graph $G \in S(k,n)$ with n pebbles and n moves. However the strategy known to achieve $O(n/\log n)$ pebbles on every graph uses exponential time. Thus it is a natural question to ask if there are graphs $G_n \in S(k,n)$ such that every strategy which achieves a minimal number of pebbles requires necessarily exponential time.This is indeed the case.

<u>Theorem:</u> There exists a family of graphs $G_n \in S(2,n)$, $n=1,2,\ldots$ and constants c_1, c_2, c_3, $c_2 < c_1$ such that for infinitely many n

1) G_n can be pebbled with $c_1\sqrt{n}$ pebbles in n moves.

2) G_n can also be pebbled with $c_2\sqrt{n}$ pebbles.

3) Every strategy which pebbles G_n using only $c_2\sqrt{n}$ pebbles has at least $2^{c_3\sqrt{n}}$ moves.

Thus even saving only a constant fraction of the pebbles already forces the time from linear to exponential.

<u>Proof of the theorem:</u> as building blocks for the graphs G_n we need certain special graphs: A <u>directed bipartite graph</u> is a graph whose nodes can be partitioned into two disjoint sets N_1, N_2 suchthat all edges go from nodes in N_1 to nodes in N_2. A directed bipartite graph is an <u>n-i/j-expander</u> if

$|N_1| = |N_2| = n$ ($|A|$ denotes the cardinality of A) and for all subsets N' of N_2 of size n/i holds:

$$|\{c \mid c \in N_1 \text{ and there is an edge from } c \text{ to a node in } N'\}| > n/j .$$

Lemma 1: For big enough n there exist n-8/2-expanders where the indegree of each node in N_2 is exactly 16.

Proof of Lemma 1: With every function $f:\{1,\ldots,cn\} \to \{1,\ldots,n\}$ we associate a bipartite graph $G_f \in S(c,2n)$ with n inputs and n outputs in the following way: The inputs and outputs are numbered from 1 to n and if $f(j) = i$ then there is an edge from input i to output $(j \bmod n)$. Different functions may produce the same graph. A function f is bad if there is a set I of $n/2$ inputs and a set O of $n/8$ outputs such that all edges into O come from I. Otherwise the function f is called good. Clearly if f is good G_f is an n-8/2-expander with the desired properties.

In order to prove the existence of a good function we prove that the fraction of bad functions to all such functions tends with growing n to zero [5,6]. There are n^{cn} functions $f:\{1,\ldots,cn\} \to \{1,\ldots,n\}$. There are $\binom{n}{n/2}\binom{n}{n/8}$ ways to choose $n/2$ inputs I and $n/8$ outputs O. For every choice of I and O there are $(n/2)^{cn/8} \cdot n^{7cn/8}$ functions f such that f is bad because in G_f all edges into O come from I. Hence there are at most $\binom{n}{n/2}\binom{n}{n/8} \cdot (n/2)^{cn/8} \cdot n^{7cn/8}$ bad functions. Thus the fraction we want to estimate is

$$\binom{n}{n/2}\binom{n}{n/8}\cdot(n/2)^{cn/8}\cdot n^{7cn/8}/n^{cn} = \binom{n}{n/2}\binom{n}{n/8}/2^{cn/8} = o(1) \quad \text{for } c \geq 16 .$$

Let E_n' be an n-8/2-expander as in lemma 1. Construct E_n from E_n' by replacing for every output node v the 16 incoming edges by a complete binary tree with 16 leaves, identifying v with the root of the tree and the ancestors of v with the leaves. Obviously $E_n \in S(2,16n)$.

Let $H_{b,d}$ be the graph consisting of d copies of E_b: $E_b^1,\ldots,E_b^d$ where for $2 \leq i \leq d$ the input nodes of E_b^i are identified with the output nodes of E_b^{i-1}. Thus $H_{b,d} \in S(2,(15d+1)b)$.

The set of output nodes of E_b^i is called the i^{th} level. The input nodes of E_b^1 form level 0.

Lemma 2: $H_{b,d}$ can be pebbled with $2b+16$ pebbles and $(15d+1)b$ moves.

Proof: We say level i is full if all nodes of level i have pebbles. The strategy is to fill the levels one after another. Each level is a cut set. Thus once a new level i has been filled all pebbles above level i can be removed. Hence at most $2b$ pebbles have to be kept on two successive levels. In the process of filling level $i+1$ if level i is full the 16 extra pebbles are used on the trees between the levels. Because all trees are disjoint except for the leaves each node is pebbled exactly once.

Lemma 3: $H_{b,d}$ can be pebbled with $4d+2$ pebbles.

Proof: The depth of a node v is the number of edges in the longest path into v. In a graph $G \in S(2,n)$ every node of depth t can be pebbled with $t+2$ pebbles (this follows easily by induction on t). Every node in $H_{b,d}$ has depth at most $4d$.

The crucial point is

Lemma 4: For all $i \in \{0,1,\dots,d\}$ holds: If C is any configuration of at most $b/8$ pebbles on $H_{b,d}$, N is any subset of level i s.t. $|N| = b/4$, and M is any sequence of moves, which starts in configuration C, uses never more than $b/8$ pebbles, and during the execution of this sequence of moves each node in N has a pebble at least once, then M has at least 2^i moves.

Proof: by induction on i. For $i=0$ there is nothing to prove. Suppose the lemma is true for $i-1$. In configuration C at most $b/8$ pebbles are on the graph. Thus for at least $b/8$ of the nodes v in N no pebble is on v nor anywhere on the tree which joins v with level $i-1$ except possibly on the leaves. Let N' be a subset of these nodes of size $b/8$ and let P be the set of nodes in level $i-1$ which are connected to N'. By construction of $H_{b,d}$: $|P| \geq b/2$. Because none of the nodes in N' nor any node of their trees have pebbles except for the leaves, during the execution of M each node in P must have a pebble at some time (possibly right at the start).

Divide the strategy M into two parts M_1, M_2 at the earliest move such that during M_1 some $b/4$ nodes of P have or have had pebbles and the remaining $b/4$ or more nodes of P have never had a pebble. For M_1 the hypothesis of the lemma applies, thus M_1 has at least 2^{i-1} moves. Because M_1 leaves at most $b/8$ pebbles on the graph and M_2 also never uses more than $b/8$ pebbles the hypothesis also applies to M_2. Hence M_2 has at least 2^{i-1} moves too and the lemma follows.

Choose b such that $4d+2 \leq b/8$, e.g. $b = 32d + 16$. Then any strategy which pebbles any $b/4$ output nodes of $H_{b,d}$ using at most $4d+2$ pebbles has at least 2^d moves. Thus for at least one of these nodes v pebbling v alone with $4d+2$ pebbles must require $2^d/(b/4) \geq 2^{(1-\varepsilon)d}$ moves as $b=O(d)$. Now $n=(15d+1)b$ is the number of nodes of $H_{b,d}$. Hence $d=O(\sqrt{n})$ and

$b=O(\sqrt{n})$ and the theorem follows.

The above construction also yields:

Corollary: There exists a family of graphs $G_n \in S(2,n)$ such that for every $\epsilon > 0$ holds:any strategy which pebbles G_n using $n^{1-\epsilon}$ pebbles has more than polynomially many moves.

Proof: Choose $G_n = H_{b,d}$ with $b=n^{1-1/\log\log n}$ and $d=O(n^{1/\log\log n})$.

An interesting open problem is: does there exist a family of graphs G_n $S(2,n)$, $n=1,2,...$ such that pebbling the graphs G_n with $O(n/\log n)$ pebbles requires more than polynomially many moves ?

References:

1 S.A. Cook: An observation on time-storage trade off. Proceedings 5th ACM-STOC 1973. 29-33

2 J. Hopcroft, W. Paul and L. Valiant: On time versus space and related problems. 16th IEEE-FOCS 1975, 57-64

3 M.S. Paterson and C.E. Hewitt: Comparative schematology. Record of Project MAC Conf. on Concurrent Systems and Parallel Computation 197o, 119-128

4 W.Paul, R.E. Tarjan and J.R. Celoni: Space bounds for a game on graphs. 8th ACM-STOC 1976, 149-16o

5 M.S. Pinsker: On the complexity of a concentrator. 7th International Teletraffic Congress, Stockholm 1973

6 N. Pippenger: Superconcentrators. Technical Report IBM Yorktown Heights 1976

7 R. Sethi: Complete register allocation problems. Proceedings 5th ACM-STOC 1973, 182-195