Theoretical Computer Science 13 (1981) 315–322
North-Holland Publishing Company

NOTE

A COMPARISON OF TWO VARIATIONS OF A PEBBLE GAME ON GRAPHS

Friedhelm MEYER AUF DER HEIDE
Faculty of Mathematics, University of Bielefeld, Fed. Rep. Germany

Communicated by M.S. Paterson
Received April 1979
Revised May 1980

Abstract. The number of pebbles used in the black [black–white] pebble game corresponds to the storage requirement of the deterministic [non-deterministic] evaluation of a straight line program. Suppose a distinguished vertex of a directed acyclic graph can be pebbled with k pebbles in the black–white pebble game. Then it can be pebbled with $k' \leq \frac{1}{2}k(k-1)+1$ pebbles in the black pebble game.

1. Introduction

This paper deals with two pebble games played on directed acyclic graphs. The black pebble game was used in [5] to derive a space-efficient simulation for time-bounded Turing machines [$\mathrm{DTIME}(t(n)) \subseteq \mathrm{DTAPE}(t(n)/\log t(n))$]. The black–white pebble game was used in [3] to show that the language of all solvable path systems, which is log-space complete in polynomial time, requires $\Omega(n^{1/4})$ space on a special computational model.

Let G be a directed acyclic graph (DAG) with vertex set V and edge set E, and U_x the set of all predecessors of x, i.e. the set of all vertices from which there is a directed path to x in G. For $x \in V$ let V_x be the set $U_x \cup \{x\}$ and G_x the induced subgraph of G with vertex set V_x.

The *black–white pebble game* is played on a DAG G by placing black or white pebbles on some vertices of G according to the following rules:

Rule 1. It is always allowed to place a white pebble on a vertex (which contains no pebble).

Rule 2. It is always allowed to remove a black pebble from a vertex.

Rule 3. If all direct predecessors of a vertex x contain a black or white pebble and x contains no pebble, then it is allowed to place a black pebble on x.

Rule 4. If all direct predecessors of a vertex x contain a black or white pebble and x contains a white pebble, then it is allowed to remove this pebble from x.

The close connection between Rule 1 and 2 respectively 3 and 4 is emphasized in Lemma 1.

A *move* of a pebble game is a placing or removing of a pebble according to one of the four rules. For technical reasons it is also a move to do nothing.

A *configuration* of G is a pair (B, W) of disjoint subsets of V. $B[W]$ is the set of all vertices on which black [white] pebbles are lying.

We say "(B, W) directly derives (B', W') using k pebbles" and write "$(B, W) \Rightarrow_k (B', W')$" iff $\#(B \cup W) \leq k$, $\#(B' \cup W') \leq k$ and (B', W') arises from (B, W) by one move.

A sequence $[(B_i, W_i), i = 1, \ldots, n]$ is called "*a b/w-k-strategy* from (B, W) to (B', W')" iff $(B_i, W_i) \Rightarrow_k (B_{i+1}, W_{i+1})$ for all $i = 1, \ldots, n-1$, $(B_1, W_1) = (B, W)$, and $(B_n, W_n) = (B', W')$.

Immediately from the rules we can conclude the following

Lemma 1. *Let* $[(B_i, W_i), i = 1, \ldots, n]$ *be a b/w-k-strategy in G. Then* $[(W_{n-i+1}, B_{n-i+1}), i = 1, \ldots, n]$ *is a b/w-k-strategy in G.*

Let us call this strategy the counter-strategy of $[(B_i, W_i), i = 1, \ldots, n]$.

The *black pebble game* is a special kind of the black–white pebble game. It only uses black pebbles. A strategy of this game is called a b-strategy $[B_i, i = 1, \ldots, n]$.

For both games the goal is to find a strategy that starts from a configuration of G with no pebbles on the graph, ends with a black pebble on a distinguished vertex r of G and no pebbles elsewhere, and uses as few pebbles as possible. Such a strategy which uses a minimum number of pebbles is called *optimal for* (G, r).

The number of pebbles used in an optimal b/w-strategy for (G, r) is called $\mathrm{Opt}(G, r)$, and that for an optimal b-strategy, $\mathrm{Opt}_b(G, r)$.

The black [black–white] pebble game can be looked upon as a model of a deterministic [non-deterministic] evaluation of a straight line program: The instructions of the program correspond to the vertices of the graph. (a, b) is an edge in the graph, if the result of a is an operand for the computation of b. Placing a black pebble on x corresponds to computing x from its predecessors (which are all pebbled) and putting it into a register. Removing a black pebble corresponds to freeing a register. Placing a white pebble on x means that we guess a value for x to be computed intending later to justify this guess. This justification corresponds to removing the white pebble. (We are able to justify the guess before removing the white pebble, because all its predecessors are available).

Thus the storage requirement of the deterministic [non-deterministic] evaluation of a straight-line program corresponds to the number of pebbles used in the black [black–white] pebble game.

Our main result will be that the storage requirement of a deterministic and a non-deterministic evaluation of a straight-line program differ by a square-root at most.

2. Results about pebble games

The black pebble game especially is considered in many papers. Here we see some results that are interesting for the comparison of both games:

(2.1) For both games, it is known that if G is a DAG with indegree 2 and n vertices, then an optimal strategy from $(\emptyset, \emptyset)$ to $(\{r\}, \emptyset)$ for some $r \in V$ uses at most $O(n/\log n)$ pebbles [1], and there exists a family of graphs which needs this number of pebbles [2, 5].

(2.2) If S_m is a pyramid with m levels and root r (S_5 is shown in Fig. 1), then $\mathrm{Opt}_b(S_m, r) = m + 1$ for $m > 1$ [4], and $\mathrm{Opt}(S_m r) \geq \sqrt{\frac{1}{2}m} - 1$ [3].

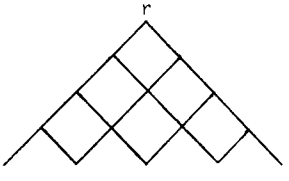

Fig. 1. The pyramid S_5.

(2.3) For an l-ary complete tree with depth n and root r, T_n^l, it is proved in [6] and [7] independently that

$$\mathrm{Opt}_b(T_n^l, r) = (n-1)(l-1) + l + 1$$

and

$$\mathrm{Opt}(T_n^l, r) = \lceil \tfrac{1}{2}(l-1)n + l + 1 \rceil + 1.$$

For trees T with root r it is shown in [6] that $\mathrm{Opt}(T, r) \geq \frac{1}{6}\mathrm{Opt}_b(T, r)$.

This result is improved in [8]. It is shown that $\mathrm{Opt}(T, r) \geq \frac{1}{2}\mathrm{Opt}_b(T, r)$.

Now we shall see that the black–white pebble game requires only half as many pebbles to pebble the top of a pyramid as the black pebble game.

Theorem 1. $\mathrm{Opt}(S_m, r) \leq \lceil \frac{1}{2}m \rceil + 2$.

Proof. By induction on m. Obviously, $\mathrm{Opt}(S_1, r) = 1$, $\mathrm{Opt}(S_2, r) = 3$.

Let $m \geq 3$ and $[C_{n-2}]$ be the $(\lceil \frac{1}{2}(m-2) \rceil + 2)$ – strategy for S_{m-2} given by induction hypothesis and $[\overline{C_{n-2}}]$ the counter-strategy of $[C_{n-2}]$ (see Lemma 1).

Then consider the following strategy: (We use the notations of Fig. 2).

(1) place a black pebble on a by $[C_{n-2}]$,

(2) place a black pebble on c by $[C_{n-2}]$,

(3) place a white pebble on b,
(4) make three moves as shown in Fig. 2,
(5) remove the white pebble from b by $[\overline{C_{n-2}}]$.
This strategy needs $\max\{([\frac{1}{2}(m-2)]+2)+1, 4\} = [\frac{1}{2}m]+2$ pebbles.

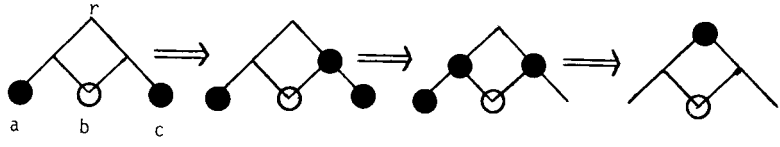

Fig. 2. The top of S_m.

The main result of this paper is the following:

Theorem 2. *Let* $G=(V,E)$ *be a* DAG, $r\in V$, $\mathrm{Opt}(G,r)=k$, *then* $\mathrm{Opt}_b(G,r)\leq \frac{1}{2}(k^2-k)+1$, *i.e.*, $\mathrm{Opt}(G,r)\geq \frac{1}{2}+\sqrt{2\,\mathrm{Opt}_b(G,r)-\frac{7}{4}}$.

With the help of (2.3) we can improve (2.2):

Corollary 1. $\mathrm{Opt}(S_m,r)\geq \frac{1}{2}+\sqrt{2m+\frac{1}{4}}$ *for* $m>1$.

In order to prove Theorem 2 we simulate a special optimal b/w-strategy, we call it a standard strategy, move by move by a b-strategy. The critical point is the simulation of a move which places a white pebble on a vertex. This simulation will be done by replacing this move by a b-strategy which places a black pebble on this vertex. The property 'standard' will guarantee that this strategy does not require too many pebbles.

3. The standard strategy

Let $[(B_i, W_i),\ i=1\ldots n]$ be a b/w-k-strategy from $(\emptyset,\emptyset)$ to $(\{r\},\emptyset)$ in the DAG $G=(V,E)$. Then the induced subgraph of G with vertex set $V_x\backslash(B_{l-1}\cup W_{l-1})$ for $x\in V$ and $1\leq l\leq n$ is called S_x^l.

Definition 1. $[(B_i, W_i), i=1\ldots n]$ is called *standard*, if the following property holds: for all $l=1\ldots n$; if in the lth move a white pebble is placed on x, then $\mathrm{Opt}(S_x^l, x)\leq k-1$.

In this chapter we shall prove that it suffices to deal with standard strategies in order to compare $\mathrm{Opt}(G,r)$ and $\mathrm{Opt}_b(G,r)$:

Main Lemma. *For every* DAG G *and vertex* r *of* G *there exists an optimal strategy which is standard.*

For the proof we consider a DAG G, a vertex r of G, and a b/w-k-strategy $[(B_i, W_i),\ i = 1 \ldots n]$ from $(\emptyset, \emptyset)$ to $(\{r\}, \emptyset)$ in G.

We present an algorithm which transforms this strategy into a new sequence $[(B_i^*, W_i^*),\ i = 1 \ldots m]$ and prove that it is a standard b/w-k-strategy from $(\emptyset, \emptyset)$ to $(\{r\}, \emptyset)$ in G.

Let $[(B_i, W_i),\ i = 1 \ldots n]$ be the input for the following algorithm.

Begin:
Let $\{l_1 \cdots l_p\}$ be the set of numbers such that

(**) $W_{l_i+1} \setminus W_{l_i} = \{x_i\}$ for some x_i
and there is a $j \geqslant l_i + 1$ such that $\#(W_j \cup B_j) \cap V_{x_i}) = k$.
Let j_i be the maximal such j and $t_i = \max\{h \mid x_i \in W_{l_i+1} \cdots W_h\}$.

Loop:
For $i = 1$ **until** p **do if** $j_i < t_i$
Comment: One move after k pebbles are the last time in V_{x_i}, the white pebble is still on x_i;
then

$$[(B_i, W_i),\ i = 1 \ldots n] \leftarrow [(B_1 \cap U_{x_i}, W_1 \cap U_{x_i}), \ldots, (B_{j_i+1} \cap U_{x_i}, W_{j_i+1} \cap U_{x_i}),$$
$$(B_{j_i+1} \cap U_{x_i}, (W_{j_i+1} \cap U_{x_i}) \cup \{x_i\}), (B_{j_i+2}, W_{j_i+2}), \ldots, (B_n, W_n)];$$

else
Comment: $t_i \leqslant j_i$, i.e. the white pebble on x is removed in the last move which reduces the number of pebbles in V_x from k to $(k-1)$ or earlier;

$$[(B_i, W_i),\ i = 1 \ldots n] \leftarrow [(B_1 \cap U_{x_i}, W_1 \cap U_{x_i}), \ldots, (B_{t_i} \cap U_{x_i}, W_{t_i} \cap U_{x_i}),$$
$$(B_{t_i+1} \cap V_{x_i}, W_{t_i+1} \cap V_{x_i}), \ldots, (B_{j_i} \cap V_{x_i}, W_{j_i} \cap V_{x_i}),$$
$$(B_{j_i+1}, W_{j_i+1}), \ldots, (B_n, W_n)];$$

End;

The following fact allows us to restrict a strategy in G to an induced subgraph of G. It follows directly from the rules of the black–white pebble game.

Fact 1. The restriction of a b/w-strategy in a DAG G to an induced subgraph H of G is a b/w-strategy in H. If $H = G_x$ for a vertex x of G, then it is also a b/w-strategy in G.

We conclude the main lemma from the following 3 propositions:

Proposition 1. *The output sequence of the pass of the loop is a b/w-k-strategy from* $(\emptyset, \emptyset)$ *to* $(r, \emptyset)$,

Proposition 2. *If a configuration* (B, W) *is inserted in the then-clause between* (B_{j+1}, W_{j+1}) *and* (B_{j+2}, W_{j+2}), *then* $\#(B \cup W) \leq k-1$ *and after the pass of the loop,* $\#((B_q \cup W_q) \cap V_x) \leq k-1$ *for all* $q \geq j+2$,

Proposition 3. *If for some y and q* $[(B_q \cap V_y, W_q \cap V_y), \ldots, (B_m \cap V_y, W_m \cap V_y)]$ *is a b/w-*$(k-1)$*-strategy, then it is still one after the pass.*

Accept these propositions for a moment. Let $[(B_i^*, W_i^*),\ i = 1 \ldots m]$ be the output-sequence of the algorithm.

Proposition 1 guarantees that it is a b/w-k-strategy. In order to verify that it is standard consider a number l such that in the lth move a white pebble is placed on x.

Then by Fact 1 it follows that

$$[((B_i^* \cap V_x) \backslash (B_l^* \cup W_l^*), (W_i^* \cap V_x) \backslash (B_l^* \cap W_l^*)),\ i = l+1, \ldots, m],$$

the restriction of the strategy on S_x^l, is a b/w-strategy in S_x^l.

Because of Proposition 2 and 3 it uses $(k-1)$ pebbles at most.

Notice that

$$(B_{l+1}^* \cap V_x) \backslash (B_l^* \cup W_l^*) = \emptyset, \qquad (W_{l+1}^* \cap V_x) \backslash (B_l^* \cup W_l^*) = \{x\},$$

$$(W_m^* \cap V_x) \backslash (B_l^* \cup W_l^*) = \emptyset,$$

and

$$(B_m^* \cap V_x) \backslash (B_l^* \cup W_l^*) = \begin{cases} \emptyset, & x \neq r, \\ r, & x = r \end{cases} \qquad (*)$$

In the case $(*)$ remove the black pebble from r in a new move.

Thus we obtain a b/w-$(k-1)$-strategy from $(\emptyset, \{x\})$ to $(\emptyset, \emptyset)$ in S_x^l. Its counter-strategy (Lemma 1) guarantees that $\mathrm{Opt}(S_x^l, x) \leq k-1$.

It remains to prove Proposition 1, 2 and 3.

Proof of Proposition 1.

Case 1: The "then-clause" is executed.

- $[(B_1 \cap U_x, W_1 \cap U_x), \ldots, (B_{j+1} \cap U_x, W_{j+1} \cap U_x)]$ is a b/w-k-strategy because of Fact 1.
- $(B_{j+1} \cap U_x,\ W_{j+1} \cap U_x) \Rightarrow_k (B_{j+1} \cap U_x,\ (W_{j+1} \cap U_x) \cup \{x\}))$, because it is always allowed to place a white pebble and because of the following.

  As $\#((B_j \cup W_j) \cap V_x) = k$, it follows that $B_j \cup W_j \subset V_x$ and that in the next move one pebble will be removed (j maximal!).

  Therefore, $\#(B_{j+1} \cup W_{j+1}) \leq k-1$ and as $x \in W_{j+1}$:

$$\#((B_{j+1} \cap U_x) \cup (W_{j+1} \cap U_x)) \leq k-2$$

and

$$\#((B_{j+1} \cap U_x) \cup (W_{j+1} \cap U_x) \cup \{x\}) \leq k-1. \qquad (3.1)$$

- $(B_{j+1} \cap U_x,\ (W_{j+1} \cap U_x) \cup \{x\}) \Rightarrow_k (B_{j+2},\ W_{j+2})$, because $B_{j+1} \cap U_x = B_{j+1}$ and $(W_{j+1} \cap U_x) \cup \{x\} = W_{j+1}$.

- $[(B_{j+2}, W_{j+2}), \ldots, (B_m, W_m)]$ is b/w-k-strategy in G.

Case 2: The 'else-clause' is executed.

- $[(B_1 \cap U_x, W_1 \cap U_x), \ldots, (B_t \cap U_x, W_t \cap U_x)]$ and $[(B_{t+1} \cap V_x, W_{t+1} \cap V_x), \ldots, (B_j \cap V_x, W_j \cap V_x)]$ are b/w-k-strategies because of Fact 1.
- $(B_t \cap U_x, W_t \cap U_x) \Rightarrow_k (B_{t+1} \cap V_x, W_{t+1} \cap V_x)$, because $B_t \cap U_x = B_{t+1} \cap V_x$ and $W_t \setminus W_{t+1} = \{x\}$, therefore: $W_{t+1} \cap V_x = W_t \cap U_x$.
- $(B_j \cap V_x, W_j \cap V_x) \Rightarrow_k (B_{j+1}, W_{j+1})$, because $B_j, W_j \subset V_x$.
- $[(B_{j+1}, W_{j+1}), \ldots, (B_m, W_m)]$ is a b/w-k-strategy.

Proof of Proposition 2.

- $\#(B \cup W) \leq k-1$, because $(B, W) = (B_{j+1} \cap U_x, (W_{j+1} \cap U_x) \cup \{x\})$ and because of (3.1).
- $\#((B_q \cup W_q) \cap V_x) \leq k-1$ for all $q \geq j+2$, because these configurations are left unaltered by the pass and j was chosen maximally.

Proof of Proposition 3. The algorithm inserts new configurations only in the 'then-clause', and in (3.1) we have seen that these new configurations always use fewer than k pebbles. If the algorithm manipulates some configuration (B, W), it never enlarges $\#(B \cup W)$. Proposition 3 follows by Fact 1 and Proposition 1.

4. Proof of Theorem 2

The following fact allows us to insert a strategy for S_x^l in a strategy for G. It follows directly from the rules of the game.

Fact 2. Let G be a DAG, (B, W) a configuration of G, $\bar{G}$ the induced subgraph of G with vertex set $V \setminus (B \cup W)$ and $[(B_i, W_i), i = 1 \ldots n]$ a b/w-k-strategy in $\bar{G}$. Then, $[(B_i \cup B, W_i \cup W), i = 1 \ldots n]$ is a b/w-$(k + \#(B \cup W))$-strategy in G.

We define $F(k) := \max\{\mathrm{Opt}_b(G, r); G, r \text{ chosen such that } \mathrm{Opt}(G, r) \leq k\}$. Because of the main Lemma it suffices to simulate a standard b/w-k-strategy from $(\emptyset, \emptyset)$ to $(\{r\}, \emptyset)$ in G by a $b - [\frac{1}{2}(k^2 - k) + 1]$-strategy from $\emptyset$ to $\{r\}$ in G.

Let $[(B_i, W_i), i = 1 \ldots n]$ be such a standard strategy and $\{l_1, \ldots, l_p\}$ the set of numbers such that in the l_ith move a white pebble is placed on x_i. The property standard guarantees that $\mathrm{Opt}(S_{x_i}^{l_i}, x_i) \leq k-1$ and therefore $\mathrm{Opt}_b(S_{x_i}, x_i) \leq F(k-1)$. The following lemma explains how to replace the l_ith move of the strategy by an optimal b-strategy from $\emptyset$ to $\{x_i\}$ in $S_{x_i}^{l_i}$.

Lemma 2. *Let $[(B_i, W_i), i = 1 \ldots n]$ be a b/w-k-strategy in G. If a white pebble is placed on x in the lth move and removed in the tth move, $\#(B_l \cup W_l) = d$, and there*

is a b–k_1-strategy $[D_i, i = 1 \ldots p]$ in S_x^l from $\emptyset$ to $\{x\}$ and $\bar{k} = \max\{d + k_1, k\}$, then

$$[(B_1, W_1), \ldots, (B_l, W_l), (B_l \cup D_1, W_l), \ldots, (B_l \cup D_p, W_l),$$
$$(B_{l+2} \cup \{x\}, w_{l+2} \backslash \{x\}), \ldots, (B_t \cup \{x\}, W_t \backslash \{x\}), (B_{t+1}, W_{t+1}), \ldots, (B_n, W_n)]$$

is a b/w-$\bar{k}$-strategy in G.

Proof. It is clear that the maximum number of pebbles used in every configuration is $\bar{k}$.

- $[(B_1, W_1), \ldots, (B_l, W_l)]$ and $[(B_{t+1}, W_{t+1}) \ldots (B_n, W_n)]$ are b/w-$\bar{k}$-strategies in G.
- $(B_l, W_l) \Rightarrow_k (B_l \cup D_1, W_l)$, because $D_1 = \emptyset$.
- $[B_l \cup D_1, W_l), \ldots, (B_l \cup D_p, W_l)]$ is a b/w-$\bar{k}$-strategy because of the main lemma.
- $(B_l \cup D_p, W_l) \Rightarrow_{\bar{k}} (B_{l+2} \cup \{x\}, W_{l+2} \backslash \{x\})$ because $D_p = \{x\}$, and therefore $B_{l+1} \cup W_{l+1} = B_l \cup D_p \cup W_l$.
- $[(B_{l+2} \cup \{x\}, W_{l+2} \backslash \{x\}, \ldots, (B_t \cup \{x\}, W_t \backslash \{x\})]$ is a b/w-$\bar{k}$-strategy because $B_{l+i} \cup W_{l+i} = (B_{l+i} \cup \{x\}) \cup (W_{l+i} \backslash \{x\})$.
- $(B_t \cup \{x\}, W_t \backslash \{x\} \Rightarrow_k (B_{t+1}, W_{t+1})$, because $W_t \backslash \{x\} = W_{t+1}$, $B_{t+1} = B_t$, and the removal of black pebbles is always allowed.

Now we execute such a replacement for every l_i. The result is a b-strategy from $\emptyset$ to $\{r\}$ in G which uses $F(k-1) + (k-1)$ pebbles at most (Notice that $\# (B_{l_i} \cup W_{l_i}) \leq (k-1)$. This simulation is possible for every DAG G and vertex r of G with $\mathrm{Opt}(G, r) \leq k$. Therefore $F(k) \leq F(k-1) + (k-1)$. As obviously $F(1) = 1$ we obtain that $F(k) \leq \frac{1}{2}(k^2 - k) + 1$.

5. Conclusion

We have seen that the number of pebbles required in the black and black–white pebble games differ by a square-root at most, but there is no family of graphs known in which it does save more than a factor $\frac{1}{2}$.

References

[1] W. Paul, R. E. Tarjan and J.R. Celoni, Space bounds for a game on graphs, *Math. Systems Theory* **10** (1976/77) 239–251.

[2] J.R. Gilbert and R.E. Tarjan, Variations of a pebble game, Report No. CS-78-661, Department of Computer Science, Stanford University (1978).

[3] S. Cook and R. Sethi, Storage requirements for deterministic polynomial time recognizable languages, *Comput. System. Sci.* **13** (1976) 25–37.

[4] S. Cook, An observation on time-storage trade off, *J. Comput. System Sci.* **9** (1974) 308–316.

[5] J. Hocroft, W. Paul and L. Valiant, On time versus space, *J. ACM* **24** (1977) 332–337.

[6] F. Meyer auf der Heide, A comparison between two variations of a pebble game on graphs, Diplomarbeit Universität Bielefeld, Fakultät für Mathematik (1979).

[7] M.C. Loui, The space complexity of two pebble games on trees, TM-133, Laboratory for Computer Science, Massachusetts Institute of Technology (1979).

[8] T. Lengauer and R.E. Tarjan, The space complexity of pebble games on trees, Preprint, Stanford (1979).