Pebbling Meets Coloring : Reversible Pebble Game On Trees

Balagopal Komarath* Jayalal Sarma Saurabh Sawlani

Department of Computer Science & Engineering,
Indian Institute of Technology Madras, Chennai - 36, India.

Email : {baluks|jayalal.sarma|saurabh.sawlani}@gmail.com

April 19, 2016

Abstract

The reversible pebble game is a combinatorial game played on rooted DAGs. This game was introduced by Bennett [1] motivated by applications in designing space efficient reversible algorithms. Recently, Siu Man Chan [2] showed that the reversible pebble game number of any DAG is the same as its Dymond-Tompa pebble number and Raz-Mckenzie pebble number.

We show, as our main result, that for any rooted directed tree T, its reversible pebble game number is always just one more than the edge rank coloring number of the underlying undirected tree U of T. The most striking implication of this result is that the reversible pebble game number of a tree does not depend upon the direction of edges, a fact that does not hold in general for DAGs. It is known that given a DAG G as input, determining its reversible pebble game number is PSPACE-hard. Our result implies that the reversible pebble game number of trees can be computed in polynomial time as edge rank coloring number of trees can be computed in linear time ([8]).

We also address the question of finding the number of steps required to optimally pebble various families of trees. It is known that trees can be pebbled in $n^{O(\log(n))}$ steps where n is the number of nodes in the tree. Using the equivalence between reversible pebble game and the Dymond-Tompa pebble game [2], we show that complete binary trees can be pebbled in $n^{O(\log\log(n))}$ steps, a substantial improvement over the naive upper bound of $n^{O(\log(n))}$.

It remains open whether complete binary trees can be pebbled in polynomial number of steps (i.e., n^k for some constant k). Towards this end, we show that *almost optimal* (i.e., within a factor of $(1 + \epsilon)$ for any constant $\epsilon > 0$) pebblings of complete binary trees can be done in polynomial number of steps.

*Sponsored by TCS Research Fellowship

We also show a time-space trade-off for reversible pebbling for families of bounded degree trees by a divide-and-conquer approach: for any constant $\epsilon > 0$, such families can be pebbled using $O(n^\epsilon)$ pebbles in $O(n)$ steps. This generalizes an analogous result of Královic[7] for chains.

1 Introduction

Pebbling games of various forms on graphs abstracts out resources in different combinatorial models of computation (See [3]). A rooted DAG can be used to model computation as follows – Each node in the DAG represents a value obtained during computation, the source nodes represent input values, the internal nodes represent intermediate values, and the root node represents the output value. A pebble placed on a vertex in a graph corresponds to storing the value at that node, and an edge (a, b) in the graph would represent a data-dependency - namely, the value at b can be computed only if the value at a is known (or stored). Devising the rules of the pebble game to capture the rules of the computation, and establishing bounds for the total number of pebbles used at any point in time, gives rise to a combinatorial approach to proving bounds on the *space* used by the computation. The Dymond-Tompa and Raz-Mckenzie pebble games depict some of the combinatorial barriers in improving upper bounds for depth (or parallel time) of Boolean circuits (or parallel algorithms).

Motivated by applications in the context of reversible computation (for example, quantum computation), Bennett[1] introduced the reversible pebble game. Given any DAG G with a unique sink node r, the reversible pebble game starts with no pebbles on G and ends with a pebble (only) on r. Pebbles can be placed or removed from any node according to the following two rules.

1. To pebble v, all in-neighbors of v must be pebbled.
2. To unpebble v, all in-neighbors of v must be pebbled.

The goal of the game is to pebble the sink node r using the minimum number of pebbles (also using the minimum number of steps).

Recently, Chan[2] showed that for any DAG G the number of pebbles required for the reversible pebble game is exactly the same as the number of pebbles required for the Dymond-Tompa pebble game and the Raz-Mckenzie pebble game. However, connections between the reversible pebble game and graph parameters not arising from computational considerations were not known. For irreversible pebble games, we know that the black white pebbling number of trees is closely related to min-cut linear arrangements of trees[13].

On the computational complexity front, Chan[2] also studied the complexity of the following problem – Given a DAG $G = (V, E)$ with a unique sink r and an integer $1 \leq k \leq |V|$, check if G can be pebbled using at most k pebbles. He showed that this problem is PSPACE-complete. Determining the irreversible black and black-white pebbling number are known to be PSPACE-complete on DAGs (See [5], [6]). If we restrict the irreversible black pebble game to be read-once (each node is pebbled only once), then the problem becomes NP-complete

(See [11]). However, if we restrict our attention to trees, the irreversible black pebble game[9] and black-white pebble game[13] are solvable in polynomial time. The key insight is that the optimal *irreversible* (black or black-white) pebbling number of trees can be achieved by read-once pebblings. Deciding whether the pebbling number is at most k for a given tree is in NP since the optimal pebbling serves as the certificate. We cannot show that determining the reversible pebbling number is in NP using the same argument as we do not know whether the optimal value can always be achieved using pebblings taking only polynomially many steps.

Our Results: In this paper, we study the reversible pebble game on trees. For an undirected tree T, the edge rank coloring number of the tree is the minimum number of colors required to color the edges of T using integers such that for any two edges in T having the same color i, there is at least one edge on the path between those edges that has a higher color. We show that the reversible pebbling number of any tree is exactly one more than the edge rank coloring number of the underlying undirected tree. Besides, the reversible pebbling number, another interesting parameter related to reversible pebble game is the number of steps required to optimally pebble the given DAG. For example, it is known that paths can be optimally pebbled in $O(n \log n)$ steps. We show that the connection with Dymond-Tompa pebble game can be exploited to show that complete binary trees have optimal pebblings that take at most $n^{O(\log\log(n))}$ steps. This is a significant improvement over the previous upper bound of $n^{O(\log(n))}$ steps. It remains open whether complete binary trees can be pebbled in polynomial number of steps. Towards this end, we show that "almost" (within a factor of $(1+\epsilon)$ for any constant $\epsilon > 0$) optimal pebblings of complete binary trees can be done in polynomial number of steps. We also generalize a time-space trade-off result given for paths by Královic to families of bounded degree trees showing that for any constant $\epsilon > 0$, such families can be pebbled using $O(n^{\epsilon})$ pebbles in $O(n)$ steps.

Complexity of Reversible Pebbling Number on Trees: We show that the reversible pebbling number of trees along with strategies achieving the optimal value can be computed in polynomial time. This is obtained by combining our main result with the linear-time algorithm given by Lam and Yue [8] for finding an optimal edge rank coloring of the underlying undirected tree. Our proof of the main result also shows how to convert an optimal edge rank coloring into an optimal reversible pebbling.

2 Preliminaries

We assume familiarity with basic definitions in graph theory, such as those found in [12]. A directed tree $T = (V, E)$ is called a *rooted directed tree* if there is an $r \in V$ such that r is reachable from every node in T. The node r is called the root of the tree.

An *edge rank coloring* of an undirected tree T with k colors $\{1, \ldots, k\}$ labels each edge of T with a color such that if two edges have the same color i, then the path between these two edges consists of an edge with some color $j > i$. The minimum number of colors required for an edge rank coloring of T is denoted by $\chi'_e(T)$.

Definition 1 *(Reversible Pebbling[1]) Let G be a rooted DAG with root r. A* reversible pebbling configuration *of G is a set $P \subseteq V$ (the set of pebbled vertices). A* reversible pebbling *of G is a sequence of reversible pebbling configurations $P = (P_1, \ldots, P_m)$ such that $P_1 = \phi$ and $P_m = \{r\}$ and for every $i, 2 \leq i \leq m$, we have*

1. *$P_i = P_{i-1} \cup \{v\}$ or $P_{i-1} = P_i \cup \{v\}$ and $P_i \neq P_{i-1}$ (Exactly one vertex is pebbled/unpebbled at each step).*

2. *All in-neighbors of v are in P_{i-1}.*

The number m is called the time taken by the pebbling P. The number of pebbles or space used in a reversible pebbling of G is the maximum number of pebbles on G at any time during the pebbling. The persistent reversible pebbling number *of G, denoted by $R^{\bullet}(G)$, is the minimum number of pebbles required to persistently pebble G.*

A closely related notion is that of visiting *reversible pebbling, where the pebbling P satisfies (1) $P_1 = P_m = \phi$ and (2) there exists a j such that $r \in P_j$. The minimum number of pebbles required for a visiting pebbling of G is denoted by $R^{\phi}(T)$.*

It is easy to see that $R^{\phi}(G) \leq R^{\bullet}(G) \leq R^{\phi}(G) + 1$ for any DAG G.

Definition 2 *(Dymond-Tompa Pebble Game [4]) Let G be a DAG with root r. A Dymond-Tompa pebble game is a two-player game on G where the two players, the pebbler and the challenger takes turns. In the first round, the pebbler pebbles the root node and the challenger challenges the root node. In each subsequent round, the pebbler pebbles a (unpebbled) node in G and the challenger either challenges the node just pebbled or re-challenges the node challenged in the previous round. The pebbler wins when the challenger challenges a node v and all in-neighbors of v are pebbled.*

The Dymond-Tompa pebble number of G, denoted $DT(G)$, is the minimum number of pebbles required by the pebbler to win against an optimal challenger play.

The Raz-Mckenzie pebble game is also a two-player pebble game played on DAGs. The optimal value is denoted by $RM(G)$. A definition for the Raz-Mckenzie pebble game can be found in [10]. Although the Dymond-Tompa game and the reversible pebble game look quite different. The following theorem reveals a surprising connection between them.

Theorem 3 *(Theorems 6 and 7, [2]) For any rooted DAG G, we have $DT(G) = R^{\bullet}(G) = RM(G)$.*

Definition 4 *(Effective Predecessor [2]) Given a pebbling configuration P of a DAG G with root r, a node v in G is called an* effective predecessor *of r iff there exists a path from v to r with no pebbles on the vertices in the path (except at r).*

Lemma 5 *(Claim 3.11, [2]) Let G be any rooted DAG. There exists an optimal pebbler strategy for the Dymond-Tompa pebble game on G such that the pebbler always pebbles an effective predecessor of the currently challenged node.*

The height or depth of a tree is defined as the maximum number of nodes in any root to leaf path. We denote by Ch_n the rooted directed path on n nodes with a leaf as the root. We denote by Bt_h the complete binary tree of height h. We use $root(Bt_h)$ to refer to the root of Bt_h. If v is any node in Bt_h, we use $left(v)$ ($right(v)$) to refer to the left (right) child of v. We use $right^i$ and $left^i$ to refer to iterated application of these functions. We use the notation $Ch_i + Bt_h$ to refer to a tree that is a chain of i nodes where the source node is the root of a Bt_h.

Definition 6 *We define the language* TREE-PEBBLE *as the set of all tuples (T, k), where T is a rooted directed tree and k is a integer satisfying $1 \leq k \leq n$, such that $R^{\bullet}(T) \leq k$. The language* TREE-VISITING-PEBBLE *is the same as* TREE-PEBBLE *except that the goal is to check whether $R^{\phi}(T) \leq k$.*

In the rest of the paper, we use the term pebbling to refer to *persistent reversible pebbling* unless explicitly stated otherwise.

3 Pebbling meets Coloring

In this section, we prove our main theorem which states that the reversible pebbling number of any tree is exactly one more than the edge rank coloring number of its underlying undirected tree. It is helpful to think about how to solve TREE-PEBBLE in polynomial time or even NP. The first attempt would be to try and use the pebbling sequence as a certificate that the input tree has low pebbling number. But, this approach fails because trees are not guaranteed to have optimal pebbling sequences of polynomial number of steps. We propose the *strategy tree* (Definition 7) as a succinct encoding of pebbling sequences. A strategy tree describes a pebbling sequence. The key property is that for any tree, there is an optimal pebbling sequence that can be described using a strategy tree (Lemma 8).

Definition 7 *(Strategy Tree) Let T be a rooted directed tree. If T only has a single node v, then any strategy tree for T only has a single node labeled v. Otherwise, we define a strategy tree for T as any tree satisfying*

1. *The root node is labelled with some edge $e = (u, v)$ in T.*
2. *The left subtree of root is a strategy tree for T_u and the right subtree is a strategy tree for $T \setminus T_u$.*

The following properties are satisfied by any strategy tree S of $T = (V, E)$.

1. Each node has 0 or 2 children.

2. There are bijections from E to internal nodes of S and from V to leaves of S.

3. Let v be any node in S. Then the subtree S_v corresponds to the subtree of T spanned by the nodes labeling the leaves of S_v. If u and v are two nodes in S such that one is not an ancestor of the other, then the subtrees in T corresponding to u and v are vertex-disjoint.

Lemma 8 *Let T be a rooted directed tree. Then $R^\bullet(T) \leq k$ if and only if there exists a strategy tree for T of depth at most k.*

Proof We prove both directions by induction on $|T|$. If T is a single node tree, then the statement is trivial.

(if) Assume that the root of a strategy tree for T of depth k is labelled by an edge (u, v) in T. The pebbler then pebbles the node u. If the challenger challenges u, the pebbler follows the strategy for T_u given by the left subtree of root. If the challenger re-challenges, the pebbler follows the strategy for $T \setminus T_u$ given by the right subtree of the root. The remaining game takes at most $k-1$ pebbles by the inductive hypothesis. Therefore, the total number of pebbles used is at most k.

(only if) Consider an upstream pebbler that uses at most k pebbles. We are going to construct a strategy tree of depth at most k. Assume that the pebbler pebbles u in the first move where $e = (u, v)$ is an edge in T. Then the root node of S is labelled e. Now we have $R^\bullet(T_u), R^\bullet(T \setminus T_u) \leq k-1$. Let the left (right) subtree be the strategy tree obtained inductively for T_u ($T \setminus T_u$). Since the pebbler is upstream, the pebbler never places a pebble outside T_u ($T \setminus T_u$) once the challenger has challenged u (the root). ■

We now introduce a new game called the matching game played on undirected trees (Definition 9). This game acts as a link between the reversible pebble game and edge rank coloring.

Definition 9 *(Matching Game) Let U be an undirected tree. Let $T_1 = U$. At each step of the matching game, we pick a matching M_i from T_i and contract all the edges in M_i to obtain the tree T_{i+1}. The game ends when T_i is a single node tree. We define the* contraction number *of U, denoted $c(U)$, as the minimum number of matchings in the matching sequence required to contract U to the single node tree.*

Lemma 10 *Let T be a rooted directed tree and let U be the underlying undirected tree for T. Then $R^\bullet(T) = k+1$ if and only if $c(U) = k$.*

Proof First, we describe how to construct a matching sequence of length k from a strategy tree S of depth $k+1$. Let the leaves of S be the level 0 nodes. For $i \geq 1$, we define the level i nodes to be the set of all nodes v in S such that one child of v has level $i-1$ and the other child of v has level at most $i-1$. Define M_i to be the set of all edges in U corresponding to level i nodes in S. We claim that $M_1, \ldots, M_k$ is a matching sequence for U. Define S_i as the set of all nodes v in S such that the parent of v has level at least $i+1$. Let $Q(i)$ be the statement "T_{i+1} is obtained from T_1 by contracting all subtrees corresponding to nodes (See Property 3) in S_i". Let $P(i)$ be the statement "M_{i+1} is a matching in T_{i+1}". We will prove $Q(0)$ and $Q(i) \implies P(i)$ and $(Q(i) \wedge P(i)) \implies Q(i+1)$. Indeed for $i = 0$, we have $Q(0)$ because $T_1 = U$ and S_0 is the set of all leaves in S or nodes in T (Property 2). To prove $Q(i) \implies P(i)$, observe that the edges of M_{i+1} correspond to nodes in S where both children are in S_i. So these edges correspond to edges in T_{i+1} (by $Q(i)$) and the fact that these edges are pairwise disjoint since no two nodes in S have a common child).

To prove that $(Q(i) \wedge P(i)) \implies Q(i+1)$, consider the tree T_{i+2} obtained by contracting M_{i+1} from T_{i+1}. Since $Q(i)$ is true, this is equivalent to contracting all subtrees corresponding to S_i and then contracting the edges in M_{i+1} from T_1. The set S_{i+1} can be obtained from S_i by adding all nodes in S corresponding to edges in M_{i+1} and then removing both children (of these newly added nodes) from S_i. This is equivalent to combining the subtrees removed from S_i using the edge joining them. This is because M_{i+1} is a matching by $P(i)$ and hence one subtree in S_i will never be combined with two other subtrees in S_i. But then contracting subtrees in S_{i+1} from T_1 is equivalent to contracting S_i followed by contracting M_{i+1}.

We now show that a matching sequence of length at most k can be converted to a strategy tree of depth at most $k+1$. We use proof by induction. If the tree T is a single node tree, then the statement is trivial. Otherwise, let e be the edge in the last matching M_k in the sequence and let (u, v) be the corresponding edge in T. Label the root of S by e and let the left (right) subtree of root of S be obtained from the matching sequence $M_1, \ldots, M_{k-1}$ restricted to T_u ($T \setminus T_u$). By the inductive hypothesis, these subtrees have height at most $k-1$. ■

Lemma 11 *For any undirected tree U, we have $c(U) = \chi'_e(U)$.*

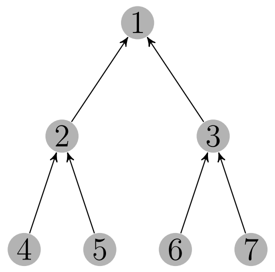
Proof Consider an optimal matching sequence for U. If the edge e is contracted in M_i, then label e with the color i. This is an edge rank coloring. Suppose for contradiction that there exists two edges e_1 and e_2 with label i such that there is no edge labelled some $j \geq i$ between them. We can assume without loss of generality that there is no edge labelled i between e_1 and e_2 since if there is one such edge, we can let e_2 to be that edge. Then e_1 and e_2 are adjacent in T_i and hence cannot belong to the same matching.

Consider an optimal edge rank coloring for U. Then in the i^{th} step all edges labelled i are contracted. This forms a matching since in between any two edges labelled i, there is an edge labelled $j > i$ and hence they are not adjacent in T_i. ■

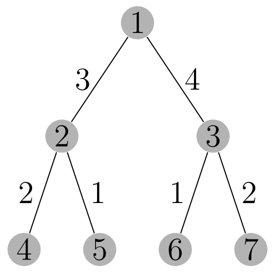
The theorems in this section are summarized in Fig. 1

Theorem 12 *Let T be a rooted directed tree and let U be the underlying undirected tree for T. Then we have $R^\bullet(T) = \chi'_e(U) + 1$.*

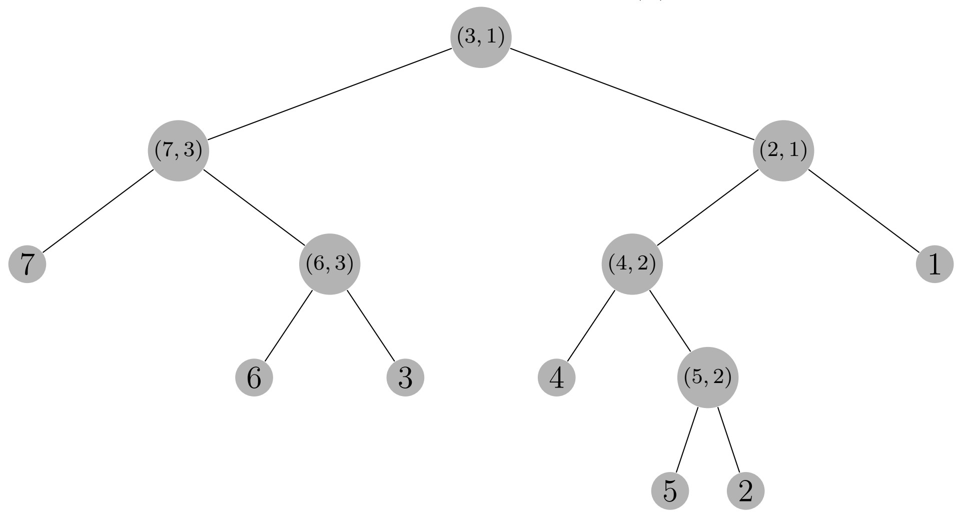
Corollary 13 *$R^\phi(T)$ and $R^\bullet(T)$ along with strategy trees achieving the optimal pebbling value can be computed in polynomial time for trees.*



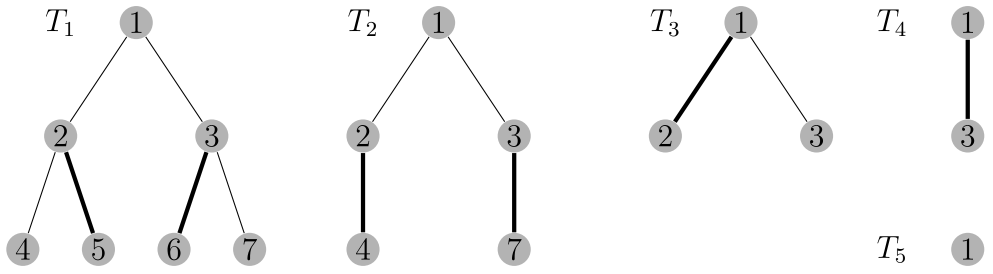
(a) The complete binary tree of height 3



(b) Optimal edge rank coloring



(c) Optimal strategy tree



(d) Optimal matching sequence

Figure 1: This figure illustrates the equivalence between persistent reversible pebbling, matching game and edge rank coloring on trees by showing an optimal strategy tree and the corresponding matching sequence and edge rank coloring for height 3 complete binary tree.

Proof We show that TREE-PEBBLE and TREE-VISITING-PEBBLE are polynomial time equivalent. Let T be an instance of TREE-PEBBLE. Pick an arbitrary leaf v of T and root the tree at v. By Theorem 12, the reversible pebbling number of this tree is the same as that of T. Let T' be the subtree rooted at the child of v. Then we have $R^{\bullet}(T) \leq k \iff R^{\phi}(T') \leq k-1$.

Let T be an instance of TREE-VISITING-PEBBLE. Let T' be the tree obtained by adding the edge (r, r') to T where r is the root of T. Then we have $R^{\phi}(T) \leq k \iff R^{\bullet}(T') \leq k+1$.

The statement of the theorem follows from Theorem 12 and the linear-time algorithm for finding an optimal edge rank coloring of trees[8]. ∎

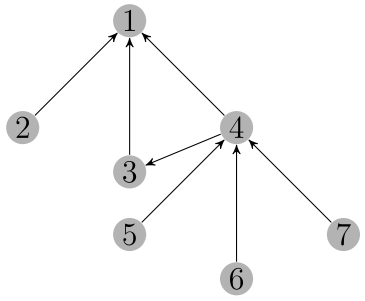
The following corollary is immediate from the equivalence of pebble games (Theorem 3).

Corollary 14 *For any rooted directed tree T, we can compute $DT(T)$ and $RM(T)$ in polynomial time.*

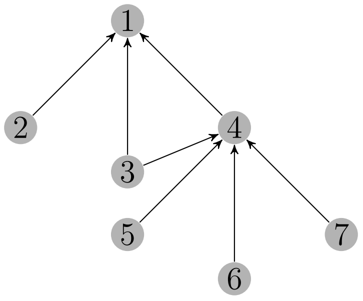
An interesting consequence of Theorem 12 is that the persistent reversible pebbling number of a tree depends only on its underlying undirected graph. A natural question would be to ask whether this fact generalizes to DAGs. The following proposition shows that this is not the case.

Proposition 15 *There exists two DAGs with the same underlying undirected graph and different pebbling numbers.*

Proof Consider the following two DAGs DAGs G_1 and G_2 have the same underlying



(a) $R^{\bullet}(G_1) = 5$



(b) $R^{\bullet}(G_2) = 6$

undirected graph and different persistent pebbling numbers. ∎

4 Time Upper-bound for an Optimal Pebbling of Complete Binary Trees

In this section, we improve time upper bounds for optimally pebbling complete binary trees. It is known that the optimal pebbling number of complete binary trees is $\log(h) + \theta(\log^*(h))$, where h is the height of the tree and $\log^*$ is the iterated logarithmic function([7]). We give an optimal pebbling of complete binary trees that takes at most $n^{O(\log\log(n))}$ steps, where n is the number of nodes in the tree. Our pebbling is essentially the same as in [7]. Our main contribution is to show that the pebbling given in [7] is optimal. This proof , like the proof of Theorem 12, uses the equivalence between the reversible pebble game and the Dymond-Tompa pebble game.

Proposition 16 *The following statements hold.*

1. $R^\bullet(Bt_h) \geq R^\bullet(Bt_{h-1}) + 1$
2. $R^\bullet(Bt_h) \geq h + 2$ *for* $h \geq 3$
3. *([1])* $R^\bullet(Ch_n) \leq \lceil \log_2(n) \rceil + 1$ *for all* n

Proof (1) In any persistent pebbling of Bt_h, consider the earliest time after pebbling the root at which one of the subtrees of the root node has $R^\phi(Bt_{h-1})$ pebbles. At this time, there is a pebble on the root and there is at least one pebble on the other subtree of the root node. So, in total, there are at least $R^\phi(Bt_{h-1}) + 2 \geq R^\bullet(Bt_{h-1}) + 1$ pebbles on the tree.

(2) Item (1) and the fact that $R^\bullet(Bt_3) = 5$.■ ■

Theorem 17 *There exists an optimal pebbling of* Bt_h *that takes at most* $n^{O(\log\log(n))}$ *steps.*

Proof We will describe an optimal upstream pebbler in a pebbler-challenger game who pebbles $root(Bt_h)$, $left(root(Bt_h))$, $left(right(root(Bt_h)))$ and so on. In general, the pebbler pebbles $left(right^{i-1}(root(Bt_h)))$ in the i^{th} step for $1 \leq i < h - \log(h)$. An upper bound on the number of steps taken by the reversible pebbling obtained from this game (which is, recursively pebble $left(right^{i-1}(root(Bt_h)))$ for $0 \leq i < h - \log(h)$ and optimally pebble the remaining tree $Ch_{h-\log(h)} + Bt_{\log(h)}$ using any algorithm) is given below. Here the term $(2h - \log(h) + 1)^{3\log(h)}$ is an upper bound on the number of different pebbling configurations with $3\log(h)$ pebbles, and therefore an upper bound for time taken for optimally pebbling the tree $Ch_{h-\log(h)} + Bt_{\log(h)}$.

$$\begin{aligned}
t(h) &\leq 2\left[t(h-1) + t(h-2) + \ldots + t(\log(h)+1)\right] + (2h - \log(h) + 1)^{3\log(h)} \\
&\leq 2ht(h-1) + (2h - \log(h) + 1)^{3\log(h)} \\
&= O\left((2h)^h (2h)^{3\log(h)}\right) \\
&= (\log(n))^{O(\log(n))} = n^{O(\log\log(n))}
\end{aligned}$$

In the first step, the pebbler will place a pebble on $left(root(Bt_h))$ and the challenger will re-challenge the root node. These moves are optimal. Before the i^{th} step, the tree has pebbles on the root and $left(right^j(root(Bt_h)))$ for $0 \leq j < i - 1$. We argue that if $i < h - \log(h)$, placing a pebble on $left(right^{i-1}(root(Bt_h)))$ is an optimal move. If the pebbler makes this move, then the cost of the game is $\max(R^\bullet(Bt_{h_1-1}), R^\bullet(Ch_i + Bt_{h_1-1})) = R^\bullet(Ch_i + Bt_{h_1-1}) \leq R^\bullet(Bt_{h_1-1}) + 1 = p$, where $h_1 = h - i + 1$. Note that the inequality here is true when $i < h - \log(h)$ by Prop 16. We consider all other possible pebble placements on i^{th} step and prove that all of them are inferior.

- *A pebble is placed on the path from the root to* $right^{i-1}(root(Bt_h))$ *(inclusive)*: The challenger will challenge the node on which this pebble is placed. The cost of this game is then at least $R^{\bullet}(Bt_{h_1}) \geq p$.
- *A pebble is placed on a node with height less than* $h_1 - 1$: The challenger will re-challenge the root node and the cost of the game is at least $R^{\bullet}(Ch_i + Bt_{h_1-1})$.

The theorem follows. For completeness, the following figure represents the optimal pebbler strategy used in the proof of Theorem 17 for proving time upper bounds for complete binary tree. ■

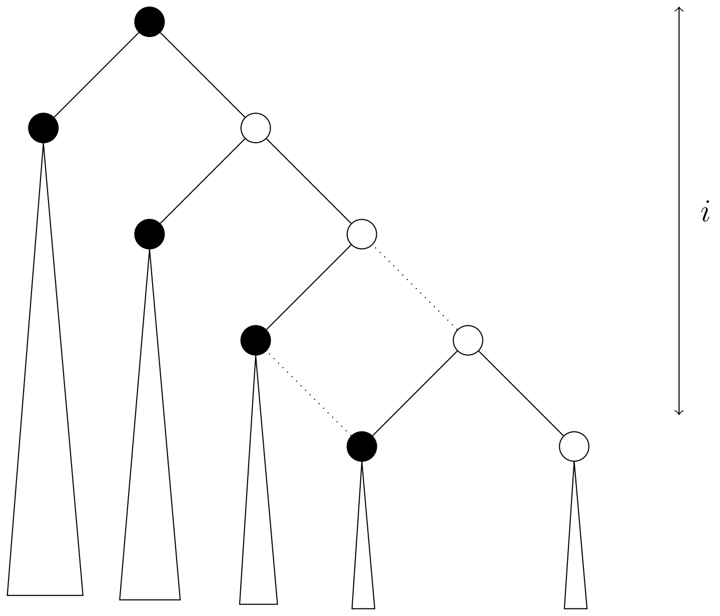


Figure 2: An Optimal Pebbling for Complete Binary Trees

5 Almost Optimal Pebblings of Complete Binary Trees

In light of Theorem 17, the natural question to ask is whether there are polynomial time optimal pebblings for complete binary trees. In this section, we show that we can get arbitrarily close to optimal pebblings for complete binary trees using a polynomial number of steps (Theorem 18).

Theorem 18 *For any constant* $\epsilon > 0$*, we can pebble* Bt_h *using at most* $(1+\epsilon)h$ *pebbles and* $n^{O(\log(1/\epsilon))}$ *steps for sufficiently large* h*.*

Proof Let $k \geq 1$ be an integer. Then consider the following pebbling strategy parameterized by k.

1. Recursively pebble the subtrees rooted at $left(right^i(root(Bt_h)))$ for $0 \leq i \leq k-1$ and $right^k(root(Bt_h))$.
2. Leaving the $(k+1)$ pebbles on the tree (from the previous step), pebble the root node using an additional k pebbles in $2k-1$ steps.
3. Retaining the pebble on the root, reverse step (1) to remove every other pebble from the tree.

The number of pebbles and the number of steps used by the above strategy on Bt_h for sufficiently large h is given by the following recurrences.

$$S(h) \leq S(h-k) + (k+1) \leq \frac{(k+1)}{k}h$$

$$T(h) \leq 2\left[\sum_{i=1}^{k} T(h-i)\right] + (2k+2) \leq (2k)^h(2k+2) \leq n^{\log(k)+1}(2k+2)$$

where n is the number of nodes in Bt_h.

If we choose $k > 1/\epsilon$, then the theorem follows. ∎

6 Time-space Trade-offs for Bounded-degree Trees

In [7], it is shown that there are linear time pebbling sequences for paths that use only n^ϵ pebbles for any constant $\epsilon > 0$. In this section, we generalize this result to bounded degree trees (Theorem 19).

Theorem 19 *For any constant positive integer k, a bounded-degree tree T consisting of n vertices can be pebbled using at most $O\left(n^{1/k}\right)$ pebbles and $O(n)$ pebbling moves.*

Proof Let us prove this by induction on the value of k. In the base case ($k = 1$), we are allowed to use $O(n)$ pebbles. So, the best strategy would to place a pebble on every vertex of T in bottom-up fashion, starting from the leaf nodes. After the root is pebbled, we unpebble each node in exactly the reverse order, while leaving the root pebbled.

In this strategy, clearly, each node is pebbled and unpebbled at most once. Hence the number of pebbling moves must be bounded by $2n$. Hence, a tree can be pebbled using $O(n)$ pebbles in $O(n)$ moves.

Now consider that for $k \leq k_0 - 1$, where k_0 is an integer ≥ 2, any bounded-degree tree T with n vertices can be pebbled using $O\left(n^{1/k}\right)$ pebbles in $O(n)$ moves. Assume that we are allowed $O\left(n^{1/k_0}\right)$ pebbles. To apply induction, we will be decomposing the tree into smaller components. We prove the following claim first.

Claim 20 *Let T' be any bounded-degree tree with $n' > n^{(k_0-1)/k_0}$ vertices and maximum degree Δ. There exists a subtree T'' of T' such that the number of vertices in T'' is at least $\lfloor n^{(k_0-1)/k_0}/2 \rfloor$ and at most $\lceil n^{(k_0-1)/k_0} \rceil$.*

Proof From the classical tree-separator theorem, we know that T' can be divided into two subtrees, where the larger subtree has between $\lfloor n'/2 \rfloor$ and $\lceil n' \cdot \dfrac{\Delta}{\Delta+1} \rceil$ vertices. The key is to recursively subdivide the tree in this way and continually choose the larger subtree. However, we need to show that in doing this we will definitely strike upon a subtree with the number of vertices within the required range. Let $T'_1, T'_2, \ldots$ be the sequence of subtrees we obtain in these iterations. Also let v_i be the number of vertices in T'_i for every i. Note that $\forall i, \lfloor v_i/2 \rfloor \leq v_{i+1} \leq \lceil v_i \cdot \dfrac{\Delta}{\Delta+1} \rceil$. Assume that j is the last iteration where $v_j > \lceil n^{(k_0-1)/k_0} \rceil$. Clearly $v_{j+1} \geq \lfloor n^{(k_0-1)/k_0}/2 \rfloor$. Also, by the definition of j, $v_{j+1} \leq \lceil n^{(k_0-1)/k_0} \rceil$. Hence the proof. ■

The final strategy will be as follows:

1. Separate the tree into $\theta(n^{1/k_0})$ connected subtrees, each containing $\theta(n^{(k_0-1)/k_0})$ vertices. Claim 20 indicates that this can always be done.

2. Let us number these subtrees in the following inductive fashion: denote by T_1, the 'lowermost' subtree, i.e. every path to the root of T_1 must originate from a leaf of T. Denote by T_i, the subtree for which every path to the root originates from either a leaf of T or the root of some T_j for $j < i$. Also, let n_i denote the number of vertices in T_i.

3. Pebble T_1 using $O\left(n_1^{1/(k_0-1)}\right) = O\left(n^{1/k_0}\right)$ pebbles. From the induction hypothesis, we know that this can be done using $O(n_1)$ pebbling moves.

4. Retaining the pebble on the root node of T_1, proceed to pebble T_2 in the same way as above. Continue this procedure till the root node of T is pebbled. Then proceed to unpebble every other vertex by executing every pebble move up to this instant in reverse order.

Now we argue the bounds on the number of pebbles and pebbling moves of the algorithm. Recall that the number of these subtrees is $O\left(n^{1/k_0}\right)$. Therefore, the number of intermediate pebbles at the root nodes of these subtrees is $O\left(n^{1/k_0}\right)$. Additionally, while pebbling the last subtree, $O\left(n^{1/k_0}\right)$ pebbles are used. Therefore, the total number of pebbles at any time remains $O\left(n^{1/k_0}\right)$. Each of the subtrees are pebbled and unpebbled once (effectively pebbled twice). Therefore the total number of pebbling moves is at most $\sum_i 2O(n_i) = O(n)$. ■

7 Discussion & Open Problems

We studied reversible pebbling on trees. Although there are polynomial time algorithms for computing black and black-white pebbling numbers for trees, it was unclear, prior to our work, whether the reversible pebbling number for trees could be computed in polynomial time. We also established that almost optimal pebbling can be done in polynomial time.

We conclude with the following open problems.

- Prove or disprove that there is an optimal pebbling for complete binary trees that takes at most $O\left(n^k\right)$ steps for a fixed k.
- Prove or disprove that the there is a constant k such that optimal pebbling for any tree takes at most $O\left(n^k\right)$ (for black and black-white pebble games, this statement is true with $k = 1$).
- Give a polynomial time algorithm for computing optimal pebblings of trees that take the smallest number of steps.

References

[1] Charles H. Bennett. Time/space trade-offs for reversible computation. *SIAM Journal of Computing*, 18(4):766–776, August 1989.

[2] Siu Man Chan. Just a pebble game. In *Proceedings of the 28th Conference on Computational Complexity, (CCC)*, pages 133–143, 2013.

[3] Siu Man Chan. *Pebble Games and Complexity.* PhD thesis, EECS Department, University of California, Berkeley, Aug 2013.

[4] Patrick W. Dymond and Martin Tompa. Speedups of deterministic machines by synchronous parallel machines. *Journal of Computer and System Sciences*, 30(2):149 – 161, 1985.

[5] John R. Gilbert, Thomas Lengauer, and Robert Endre Tarjan. The pebbling problem is complete in polynomial space. *SIAM Journal on Computing*, 9(3):513–524, 1980.

[6] Philipp Hertel and Toniann Pitassi. The pspace-completeness of black-white pebbling. *SIAM J. Comput.*, 39(6):2622–2682, April 2010.

[7] Richard Královic. Time and space complexity of reversible pebbling. In Leszek Pacholski and Peter Ruzicka, editors, *SOFSEM 2001: Theory and Practice of Informatics*, volume 2234 of *Lecture Notes in Computer Science*, pages 292–303. 2001.

[8] Tak Wah Lam and Fung Ling Yue. Optimal edge ranking of trees in linear time. In *Proc. of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 436–445, 1998.

[9] Michael C Loui. The space complexity of two pebbles games on trees. Technical Report MIT/LCS/TM-133, Massachusetts Institute of Technology, 1979.

[10] Ran Raz and Pierre McKenzie. Separation of the monotone NC hierarchy. *Combinatorica*, 19(3):403–435, 1999. Conference version appeared in proceedings of 38th Annual Symposium on Foundations of Computer Science (FOCS 1997, Pages 234–243).

[11] Ravi Sethi. Complete register allocation problems. *SIAM Journal on Computing*, pages 226–248, 1975.

[12] Douglas B. West. *Introduction to Graph Theory.* Prentice Hall, 2 edition, September 2000.

[13] Mihalis Yannakakis. A polynomial algorithm for the min-cut linear arrangement of trees. *Journal of the ACM*, 32(4):950–988, October 1985.