Supplementary Material
Moving Trains Like Pebbles: a Feasibility Study on Tree Yards

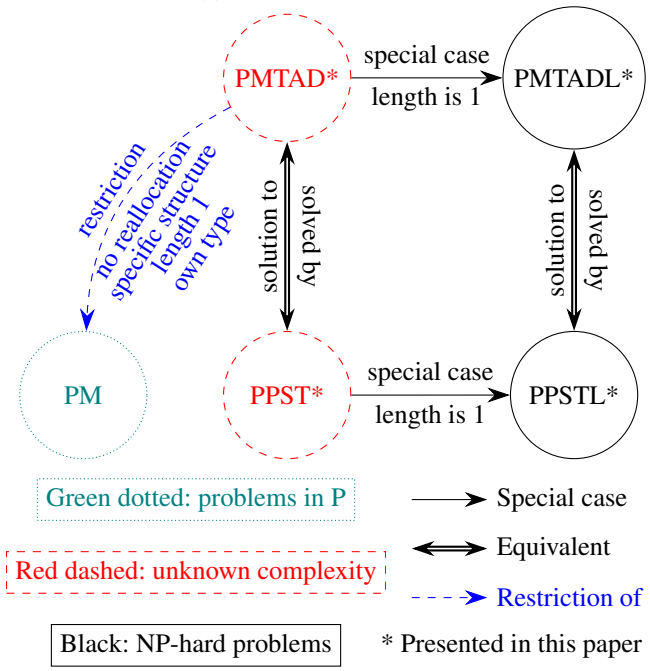
Issa K. Hanou, [1] **Mathijs M. de Weerdt,** [1] **Jesse Mulderij** [1]

[1]Delft University of Technology

1 Relation between problem variants

Short	Name
PM	Pebble Motion problem
PMTAD	Pebble Motion on a Tree with Arrival and Departure
PPST	Partition for a Pebble Sequence on a Tree
PMTADL	Pebble Motion on a Tree with Arrival and Departure, and Length inclusion
PPSTL	Partition for a Pebble Sequence on a Tree with Length inclusion

(a) Problem abbreviations.



(b) Relations between the studied problems.

Figure 1: Overview of studied problems.

2 2-PPST problem is in class P

The proof below is shown for $n = 2m$ but can be altered to also show for $n \leq 2m$ since not all branches have to be filled with pebbles.

Theorem 1 (2-PPST is in P)**.** *The Partition for a Pebble Sequence on a Tree problem is in P if there are $m = |B(T)|$ branches of length $\ell_{max} = 2$ and $n \leq 2m$ pebbles.*

Proof. Given is an instance $I = (B, P, S, D)$ of PPST, such that $m = |B(T)|$ and $n = 2m = |P|$. A $DAG(S)^+$ of the sequence S can be constructed in polynomial time. The $DAG(S)^+$ can be converted into a bipartite matching problem by creating a node $v^\flat$ and $v^\sharp$ for every $v \in DAG(S)^+$. Then, an undirected edge $\{u^\flat, v^\sharp\}$ is added for every edge $(u, v) \in DAG(S)^+$. If a matching of size m exists in the bipartite graph, then a partition $\Pi(S)$ with m tosets of size 2 can be created. Next, it is shown that I is feasible if ($\Longrightarrow$) and only if ($\Longleftarrow$) a matching of size m exists.

PROOF OF ($\Longrightarrow$): Suppose a matching of size m exists. By construction of the $DAG(S)^+$, an edge (u, v) can only exist if the pebbles associated with u and v can be in the same toset, so the tosets respect the order (condition *iii* of *partition*). Because no two edges in the bipartite matching may share an endpoint, all tosets are disjoint (condition *i* of *partition*). Furthermore, there are $2m$ nodes on each side of the bipartite graph, so a matching of size m means that exactly all nodes are covered (condition *ii* of *partition*). Because there are m tosets and m branches that are all of length $\ell_{max} = 2$, the branch set and partition are also *pairwise comparable*. So, I is feasible.

PROOF OF ($\Longleftarrow$): Suppose I is feasible. Then, a partition $\Pi(S)$ exists with m tosets of size 2 that fit on m branches of length $\ell_{max} = 2$. So, a bipartite matching between the pebbles can be created, which will make m pairs of two pebbles. □

3 PPSTL is NP-complete

Theorem 3 (PPSTL is NP-complete)**.** *The Partition for a Pebble Sequence on a Tree with Length inclusion problem is NP-complete.*

Proof. PPSTL $\in$ NP: Given a solution $\Pi(S)$ to an instance $I' = (B, P, S, D, \ell, \lambda)$ of the PPSTL problem, it can be established in $O(n)$ time whether the solution is a valid so-

lution of the PPSTL by checking all conditions of a valid partition and the pairwise comparability in length.

PPSTL $\in$ NP-HARD: Given an instance $I = (X)$ of the classic Partition Problem, an instance $I' = (B, P, S, D, \ell, \lambda)$ of the PPSTL is constructed by creating $m = 2$ branches in B with each $n - 1$ nodes and a length $\ell(b) = \frac{\sum X}{2}$. Then, a pebble $p_x \in P$ with size $\lambda(p_x) = x$ is created for every element $x \in X$ and all pebbles are added to the sequences S and D in non-descending order: $S(p_x) < S(p_y), D(p_x) < D(p_y) : \forall x, y \in X$ s.t. $x < y$.

All these steps can be done in $O(n)$ time, so the reduction is polynomial. Next, it is shown that the I is a yes-instance of the PP if ($\Longrightarrow$) and only if ($\Longleftarrow$) I' is a yes-instance of the PPSTL.

PROOF OF ($\Longrightarrow$): Suppose I is a yes-instance of the classic Partition Problem, then there is a partition of the items in X over the subsets X_1 and X_2 such that $|X_1| = |X_2|$. Then, the pebbles associated with the items assigned to X_1 form one toset, and the other pebbles form a different toset. First, a valid *partition* is constructed. By definition of the PP, these tosets are disjoint (condition *i*) and their union includes all pebbles (condition *ii*). Moreover, by construction, the order of the pebbles is maintained in the tosets (condition *iii*). Next, the lengthwise *pairwise comparability* is shown. There are now two tosets and there are two branches in the created instance I', so $|\Pi(S)| \leq |B|$ (condition *i*). Finally, there are always enough nodes in the branch for the pebbles in a toset (condition *ii*), and the capacity of the branches is never exceeded (condition *iii*). Therefore, the created partition is valid and pairwise comparable in length with B.

PROOF OF ($\Longleftarrow$): Suppose I' is a yes-instance of the PPSTL, then there is a partition of pebbles into tosets $\Pi_1, .., \Pi_{|B|}$ that is *lengthwise pairwise comparable* with the individual branches of B. By construction, there are $|B| = 2$ branches, with each a length of $\frac{\sum X}{2}$. So, both tosets contain pebbles whose sizes sum up to exactly $\frac{\sum X}{2}$. By construction, each pebble with a size λ was created from the element λ. Given that there are $|\pi_1|$(resp, $|\pi_2|$) pebbles assigned to a toset π_1(resp, π_2), the $|\pi_1|$(resp, $|\pi_2|$) elements from X that correspond to these pebble sizes can be added to the subset of X_1(resp, X_2). There are no more elements than those that sum up to $\frac{\sum X}{2}$ and each element of X can be in X_1 or X_2, but not in both. So, I' was created from yes-instance I of the classic Partition Problem. □

4 The PPST and the PMTAD

Lemma 1 (Relation between PPST and PMTAD)**.** *Let an instance $I = (T_L, P, S, D)$ of the PMTAD problem and an instance $J = (B(T), P, S, D)$ of the PPST problem be given, such that $B(T)$ is a branch set of the tree T of T_L. The instance I is feasible if and only if the instance J is feasible.*

Proof. PROOF OF ($\Longrightarrow$): Suppose that I is feasible. Next, it is shown that this implies that there is a $\Pi(S)$ that must be Definition **??** with $B(T)$ by conditions *i)* and *ii)*.

First, to park each toset in a different branch $|\Pi(S)| \leq |B(T)|$ is required. In the general case, there are no two tosets, say π and ρ, that can be parked on the same branch. Only if for each pebble p_k in π it is true that $p_k < q_1, q_1 = \min_{q \in \rho}$, then all the pebbles of π can be parked in a branch, and all the pebbles of ρ can be parked above these pebbles in the same branch. This also implies that the two tosets can be merged into $R = (\pi, \rho)$. However, in general, this is not assumed to be true, so parking two arbitrary tosets π and ρ on the same branch would result in a conflict. From the definition of a partition, there would be a pebble $p \in \pi$ that arrives later than the pebbles in ρ, while p also has to depart later than all the pebbles in ρ. This means p will be blocking the pebbles from ρ, so a conflict arises, and thus, each toset requires its own branch for parking its pebbles and thus require condition *i)*.

Because both the partition and branch set are ordered by the non-ascending size of the tosets and branches respectively, they can be pairwise compared. Take the largest toset π_1 and park it in the largest branch, which can only fit if $|\pi_i| \leq |b_i|$, where b_i is the set of nodes associated with the largest branch. Because $|\Pi(S)| \leq |B(T)|$, there must be a branch for each toset, and thus, if I is feasible this implies that this branch is large enough to fit all pebbles of the toset condition *ii)*.

So, both conditions *i)* and *ii)* are shown to hold if I is feasible, so J is also feasible.

PROOF OF ($\Longleftarrow$): Suppose J is feasible, so there is a partition that matches the set of branches $B(T)$. There are $|B(T)|$ branches in the tree T and each branch b_i provides enough space to park $|b_i|$ pebbles. So, a toset π_j of size $|\pi_j| \leq |b_i|$ can park in the branch b_i. Up to $|B(T)|$ different tosets can be parked in branch b_i, in other words, if $|\Pi(S)| \leq |B(T)|$, then each toset can be parked on a separate branch (condition *i)*. Now, take the set $B(T)$ to be ordered in non-ascending order, and take a partition $\Pi(S)$ that is ordered in non-ascending order according to the toset size, then each toset can be matched pairwise to a branch which ensures condition *ii)*. So, if there exists a partition $\Pi(S)$ that meets conditions *i)* and *ii)*, then the I is feasible. □

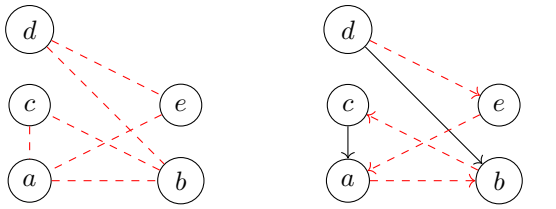
From the proof of Lemma 1, the following Corollary 1 is derived on merging tosets.

Corollary 1 (Merging tosets)**.** *Two tosets π and ρ can be merged into toset $R = (\pi, \rho)$ if and only if $p_k < q_1, \forall p_k \in \pi, q_1 = \min_{q \in \rho}$.*

Lemma 1 showed that when a partition can be found that matches the tree, the PMTAD instance is feasible. When each branch is a simple path, which is a direct child from the root of the tree, then this is both a necessary and sufficient condition. However, if the branches are not simple paths, then it is only a sufficient condition since the nested branching nodes could provide space for more feasible partitions to exist.

If there exists a partition $\Pi(S)$ that is smaller than the set of branches $B(T)$ but there is no matching such that each toset is *pairwise comparable* with a branch, then it might still be possible to feasibly park the pebbles. This is explored Lemma 2.

Lemma 2 (Combining branches for a partition)**.** *Let an instance $I = (T_L, P, S, D)$ of the PMTAD problem and a*



(a) General graph (b) Directed Acyclic Graph

Figure 2: Longest Path problem

branch set $B(T)$ be given. If there exists a strictly smaller partition $\Pi(S)$ ($|\Pi(S)| < |B(T)|$) such that its tosets are pairwise comparable with the disjoint subsets of branches $B_i \subset B(T)$, then the instance I is feasible.

Proof. Suppose there exists a set of branches $B(T)$ and a sequence of pebbles S expressed by $\Pi(S)$ such that $|\Pi(S)| < |B(T)|$. Both sets are ordered in non-descending order of item size to be able to do a pairwise comparison. Now, take the smallest toset π_1 and park its pebbles on the smallest possible branch b that is large enough, i.e. $|\pi_1| \leq |b|$, if such a branch exists. Otherwise, combine the first m branches in the ordered set $B(T)_{\leq}$ to form $B_1 \subset B(T), B_1 = \{b_1, ..b_m\}$ such that $|\pi| \leq \sum_{b \in B_1} |b|$. Not all the pebbles of one toset have to park on the same branch as long as there are no pebbles from two different tosets parked on the same branch because pebbles of the same toset cannot block each other by definition. Thus, the subsets of branches each have to be disjoint with the other subsets. Then, continue assigning tosets to a branch or a subset of branches that can hold all the pebbles of the toset. If this is possible for every toset, then the instance is feasible. □

5 Problems

5.1 Longest Path problem

Problem: Longest Path problem

Input: $I = (G)$: given is a graph $G = (V, E)$.

Question: Find the simple path in G including the maximum number of edges in E.

Consider the instance of the Longest Path problem below. On a general graph, the edges can be followed endlessly, and thus the path never really ends because there are cycles (Figure 2a). However, on the DAG in Figure 2b, the options are more limited, and the red dashed path $d \rightarrow e \rightarrow a \rightarrow b \rightarrow c$ can be constructed that is of the maximal length. Moreover, in a DAG, the Longest Path can be found in linear time (Sedgewick 2011).

5.2 Vertex-Disjoint Path Cover problem

Problem: Vertex-Disjoint Path Cover problem

Input: $I = (G, K)$: given is a graph $G = (V, E)$ and an integer K.

Question: Is there a set of K paths $\mathcal{P}$ in G such that every vertex in V belongs to exactly one path?

The problem is illustrated below, which shows two solutions for the DAG in Figure 3b: i) the red dashed paths together: $\{(d, e), (c, a, b)\}$, and ii) the black straight paths together: $\{(d, b, c), (e, a)\}$. However, for general graphs, this is more complicated. This is most clear considering the case $K = 1$ (red dashed path in Figure 3a), then the problem becomes the Hamiltonian Path problem in which the goal is to determine whether a path exists that visits every vertex exactly once. This is one of the most well-known $\mathcal{NP}$-*complete* graph problems (Sipser 2013).

For a DAG, the problem is in $\mathcal{P}$ and can be solved by transforming the problem into a bipartite graph and solving its matching problem. First, create two nodes $v^\flat, v^\sharp$ in the bipartite graph for each v in the DAG, and add an undirected edge between $u^\flat$ and $v^\sharp$ for every edge (u, v) in the DAG. Now, a matching can be found in this bipartite graph of size $|V| - K$ that corresponds to a path cover of size K, and this is possible in polynomial time (Erickson 2019).

6 Branches

Lemma 3 (Minimal number of branches)**.** *The minimal number of branches $|B(T)|$ for I to be feasible is the number of nodes in the longest path $L(S)^-$ in the negative graph $DAG(S)^-$.*

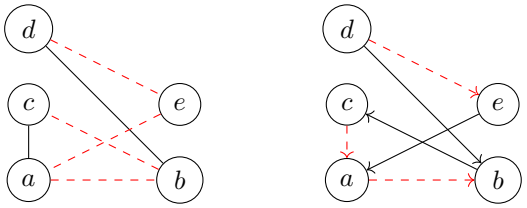
Proof. Given is the negative graph $DAG(S)^-$ of a sequence S. Take the longest path through this graph. By construction of the $DAG(S)^-$, for each pair of consecutive pebbles (p_i, p_j) along the path, so p_i arrives before p_j and departs earlier, thus p_i and p_j cannot be parked in the same branch. Otherwise, p_j arrives later and then blocks p_i for departure. Therefore, each of the pebbles corresponding to the nodes in the longest path must be parked in a different branch, so the number of nodes in this path represents the number of necessary branches in T for I to be feasible. □

This proof results in the following corollary on the infeasibility of a sequence.

Corollary 2 (Length of the longest path)**.** *If there are fewer branches than the length of the longest path $L(S)^-$ in the $DAG(S)^-$, i.e. $|B(T)| < |L(S)^-|$, then I is infeasible.*

7 Largest branches

When there are fewer branches than there are pebbles, as much space as possible should be used. Moreover, if there are exactly n nodes in the union of the branches of T,



(a) General graph with $K = 1$ (b) DAG with $K = 2$

Figure 3: Vertex-Disjoint Path Cover Problem

then precisely every node must be used. Therefore, take the largest branch of length $\ell_L = \max_{b \in B(T)} |b|$ and if there exists a toset π of at least size ℓ_L, then a pebble from π can be parked on each node in the branch. The importance of this case is shown in Lemma 4.

Lemma 4 (Largest branch)**.** *Let the tree T defined by branches $B(T)$ have $\sum_{b \in B(T)} |b| = n$, if there is no partition $\Pi(S)$ with a toset $\pi \in \Pi(S)$ such that $|\pi| \geq \ell_L = \max_{b \in B(T)} |b|$ then the instance I is infeasible.*

Proof. Suppose that the branch set $B(T)$ has n nodes in total such that n pebbles can park there. Since there are n nodes available for parking, every node in the tree must be used for parking a pebble for the instance to be feasible. So, the largest branch of length ℓ_L must thus be filled with ℓ_L pebbles. If there is no partition $\Pi(S)$ with a $\pi \in \Pi(S)$ such that $|\pi| \geq \ell_L$, then the largest branch cannot be filled with ℓ_L pebbles of the same toset. So, there will be at least one pebble parked in the largest branch that blocks another pebble. Therefore, the instance is infeasible. □

The largest possible toset can be found by taking the Longest path of the $DAG(S)^+$ constructed by the method in Method **??**. This result is used in Corollary 3.

Corollary 3 (Parking in largest branch)**.** *If $\sum_{b \in B} |b| = n$ and the length of the longest path $L(S)^+$ in the $DAG(S)^+$ is smaller than the length of the largest branch, i.e. $|B(T)| > |L(S)^+|$, then the instance is infeasible.*

This result can also be extended with empty nodes in T, which can then be left empty in the largest branch, but the rest must still be filled up. Take $c \geq 0$ to be the number of empty nodes in the tree (Corollary 4).

Corollary 4 (Parking with abundance)**.** *If $\sum_{b \in B} |b| = n + c$ and $|B(T)| > |L(S)^+| + c$, where $c \geq 0$ is the number of empty nodes in the tree, then the instance is infeasible.*

8 Partition in DAG

Method: Finding a partition in a DAG

1. Given is a directed acyclic graph $G = (V, E)$ and a path cover $C \subseteq E$
2. For each path in C, create a set of nodes $U \subseteq V$
3. For each set of nodes U, take the pebbles associated with these nodes and add them to a toset π
4. Create a partition of the tosets

Lemma 5 (Finding a partition in a DAG)**.** *Given the positive graph $DAG(S)^+$, a valid partition $\Pi(S)$ can be constructed by taking a vertex-disjoint path cover of $DAG(S)^+$.*

Proof. Given is the positive graph $DAG(S)^+$ of a sequence S. Take a path cover C and construct the partition by adding the pebbles associated with the nodes of a path together in a toset. By definition, the set of paths includes all the nodes in the graph, so the union of the tosets is P (condition *ii* of *partition*). Since C is a vertex-disjoint path cover, the tosets in $\Pi(S)$ are disjoint (condition *i* of *partition*). Finally, because the $DAG(S)^+$ respects the order of the pebbles in S by construction, the pebbles in a toset are totally ordered (condition *iii* of *partition*). So, the partition based on a vertex-disjoint path cover is a valid partition for S. □

Based on Lemma 5, the following two corollaries on specifying the partition are derived. The latter can be achieved by selecting the specific edge for the path cover, and deleting all other edges connected to the nodes in the $DAG(S)^+$ of these pebbles, so they are no longer considered as other possible edges of the path cover.

Corollary 5 (Find a vertex-disjoint path cover)**.** *Given the number of branches $|B(T)|$, a vertex-disjoint path cover of size $K = |B(T)|$ can be found.*

References

Erickson, J. 2019. *Algorithms*. University of Illinois.

Sedgewick, K. D., Robert; Wayne. 2011. *Algorithms*. Addison-Wesley Professional, 4th edition.

Sipser, M. 2013. *Introduction to the Theory of Computation, Third Edition*. Boston, MA, USA: Cengage Learning.