Theoretical Computer Science 866 (2021) 1–13

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Realization problems on reachability sequences

Matthew Dippel [1], Ravi Sundaram [1], Akshar Varma *,[1]

Northeastern University, USA

ARTICLE INFO

Article history:
Received 1 October 2020
Accepted 15 February 2021
Available online 24 March 2021

Keywords:
Reachability sequences
Graph realization
Bicriteria approximation
Strong NP-completeness

ABSTRACT

The classical Erdös-Gallai theorem (1960) kicked off the study of graph realizability by characterizing degree sequences. We extend this line of research by investigating realizability of directed acyclic graphs (DAGs) given a sequence of tuples each containing multiple node properties including the degree, reachability value (number of nodes reachable from a given node), depth and height of a node. The most interesting problems are when the sequences contain both a local constraint via degree values and a global constraint via reachability values. We show that, without degree constraints, DAG reachability realization is solvable in linear time, whereas it is strongly NP-complete given upper bounds on in-degree or out-degree. After defining a suitable notion of bicriteria approximation based on consistency, we give two approximation algorithms achieving $O(\log n)$-reachability consistency and $O(\log n)$-degree consistency; the first, randomized, uses LP (Linear Program) rounding, while the second, deterministic, employs a k-set packing heuristic. We end with some future directions of research and a set of conjectures that we hope will motivate further study of realizability with reachability constraints.

1. Introduction

Given a property P, the Graph Realization problem asks whether there exists a graph that satisfies the property P. Starting with the Erdös-Gallai paper [1] on degree sequences [2,3] many other properties have been considered in the literature ranging from eccentricities [4,5] to connectivity and flow [6,7]. The best studied among these remain extensions of realization given degree sequences [8,9] and variants focusing on different subclasses of graphs [10–19]. In addition to their theoretical significance, realization questions occur naturally in numerous application contexts, including network design [7], social networks [20,21], DNA sequencing [22], enumerating chemical compounds [23], and phylogeny and evolutionary tree reconstruction [24,25].

We consider the realization problem on digraphs[2]: we are given as input a sequence of tuples $(r_i, I_G(i), O_G(i))$, where r_i is the reachability value,[3] $I_G(i)$ the in-degree and $O_G(i)$ the out-degree of each node $i, 1 \leq i \leq n$ in an n node graph G; and we wish to determine the existence of a digraph such that each node has the prescribed reachability value and the prescribed in-degree and out-degree. This formulation extends the local properties considered by degree sequences to global properties captured by the reachability sequence. Like all realization problems this has connections [8] to the

* Corresponding author.
E-mail addresses: mdippel@ccs.neu.edu (M. Dippel), r.sundaram@northeastern.edu (R. Sundaram), varma.ak@northeastern.edu (A. Varma).

[1] This work was supported by the NSF-CNS-1718286 and NSF-CCF-1535929 grants.

[2] We only consider directed graphs and refer to them simply as graphs hereafter.

[3] The reachability value of a node is the number of nodes reachable from that node.

https://doi.org/10.1016/j.tcs.2021.02.034
0304-3975/

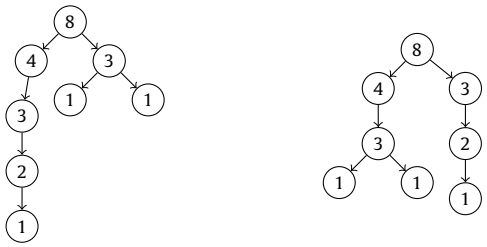

Fig. 1. Two non-isomorphic trees for the reachability sequence $S = \{1, 1, 1, 2, 3, 3, 4, 8\}$.

Table 1
Reachability realization for unbounded out-degree DAGs is linear time. The other three cases are strongly NP-complete with bicriteria approximation algorithms achieving an approximation factor of $(O(\log n), O(\log n))$.

		Out-degree	
		Bounded	Unbounded
In-Degree	Bounded (Trees)	$(O(\log n), O(\log n))$	$(O(\log n), O(\log n))$
	Unbounded (DAGs)	$(O(\log n), O(\log n))$	Linear-time

graph isomorphism problem and graph canonization. Apart from realization problems involving reachability values we also consider problems where instead of or along with the reachability values, we are also given either or both of the depth and height of nodes. Note that these are also global constraints similar to the reachability value, however we find that unlike reachability sequences depth and height realization are more tractable.

The study of reachability sequences has applications in several contexts. In the scientific context, reachability and degree constraints can reflect measurements obtained from naturally occurring networks with the aim being to generate a model that explains the measurements. Alternatively, from an engineering perspective, the goal may be to find an implementation satisfying the desired properties as specified by the reachability sequences. As an example scenario, consider the spread of a malicious virus in a network. Perhaps the first step to preventing the spread of this disease may be to understand the reach of infected nodes (using reachability values) before controlling the spread by disconnecting infected nodes from neighbors (using degree information). Alternatively, the engineering goal may be to construct resilient networks that restrict the spread of the virus.

Example 1. Does there exist a rooted tree whose reachability sequence equals the input sequence, $S = \{1, 1, 1, 2, 3, 3, 4, 8\}$, with each out-degree bounded by two?

Note that this sequence can be realized by two non-isomorphic trees. Both will contain $A = \{1, 2, 3\}$ and $B = \{1, 1, 3\}$ as subtrees but in one of the trees, A will be a child of 8 and B a child of 4 while in the other case it will be vice versa. (See Fig. 1.)

1.1. Our contributions

We characterize the complexity of the realizability of acyclic digraphs given sequences involving reachability values as summarized in Table 1. Instead of referring to bounded vs. unbounded in-degree, we simply talk about trees vs. DAGs since trees are DAGs with in-degree bounded by one and they capture the essential behavior of bounded in-degree digraphs. This usage makes the hardness results' exposition more natural and is used henceforth.

- In Section 3.2 we give linear time verifiable, necessary and sufficient conditions, for realizing unbounded out-degree DAGs (Theorem 10).
- We define a notion of bicriteria approximation in Section 2 based on local consistency of the hierarchy structure and give two algorithms in Section 4 to solve the reachability realization problem, both achieving an $(O(\log n), O(\log n))$-approximation.
  - Theorem 14: Randomized LP rounding algorithm that runs in time $O(n^{\frac{37}{18}})$ if the matrix multiplication exponent $\omega \approx 2$ and its dual $\alpha \approx 1$ [26].
  - Theorem 16: Deterministic algorithm using k-set packing heuristics [27,28] that runs in $n^{O(k^3)}$ time.
- In Section 5 we prove the strong NP-completeness of reachability realization when there are degree constraints. This includes one of our most technically involved results, a reduction from a generalized version of 3-Partition to show the strong NP-completeness of reachability realization when both in-degree and out-degree are bounded (Theorem 21). We also give a simpler reduction from the 3-Partition problem for reachability realization when only one of in-degree or out-degree is bounded (Theorems 22, 24).

- For height and depth realization, we give polynomial time algorithms for tree realization when the input sequences only involve depth (Section 3.3) and/or height (Section 3.4). Further, as corollaries to our hardness results (in Section 5.6), we show that the reductions involving reachability sequences can be easily extended to sequences where reachability values and height and/or depth are given as input. Thus, the realization problems of height/depth alongside reachability are hard while without reachability, height and depth realization is easy.

Both approximation algorithms work in the presence of non-uniform degree bounds, that is, each degree might be a different value. On the other hand, our hardness results, except Theorem 24, prove that reachability realization problems are strongly NP-complete even when the degree bounds are uniform. Specifically, Theorems 21 and 22 which have uniform degree bounds rely on having the in-degree bounded while in Theorem 24, where only the out-degree is bounded, we are only able to show hardness in the non-uniform case.

1.2. Related work

The realization problems that we study are similar in flavor to wide range of well studied problems in graph theory. The most famous is the Erdos-Gallai graph realization problem [1] (a variant also addresses the realization problem for trees) which asks whether a given set of natural numbers occurs as the degree sequence of some graph; polynomial time algorithms are known for this problem [2,3]. On the other hand it is known that realizability of DAGs is NP-Complete [13] with polynomial algorithms [11,29] for special cases.

Our problem can be considered a part of the class of well researched problems of reconstruction of a combinatorial structure from some form of partial information. Apart from the Erdos-Gallai theorem we already mentioned, a lot of work has been done on reconstruction of graphs [12,14,17–19] including collections of subgraphs [15,30] and realizability and uniqueness of graphs given invariants [8]. Beyond graphs, the problem of reconstruction of combinatorial structures like tournaments [16], trees [31,32] and matrices [33] have also been studied. More recent work has been done on reconstruction of sequences [34] and on reconstruction of strings from substrings [35].

Bartha and Bursci have addressed the problem of reconstruction of trees from frequencies of sub-trees sizes [36]. While their paper focuses on the reconstruction of unrooted trees given number of sub-trees of a given size, we look at the existence of rooted trees with a given sub-tree sequence. Our work is also different from works on phylogeny reconstruction [24,37] as those preserve distance measures between nodes while works on evolutionary trees [25] want subtrees homeomorphic to a given labeling of leaves.

2. Preliminaries

Let n denote the length of the given reachability sequence and V the set of nodes in the corresponding graph, so that $|V| = n$. For node i in graph G we let $C(i)$ denote the set of children of i, i.e., $C(i) = \{j | (i, j) \in G\}$.

Definition 1 *(Degree).* The **out-degree** of i, $O_G(i)$ is the number of its children, i.e., $|C(i)|$; the **in-degree** of node i in graph G, $I_G(i)$ is the number of nodes with arcs directed into i.

Definition 2 *(Reachability).* The reachability value of node u is the number of nodes it can reach: $r_u = |\{v : \exists \text{ path from } u \text{ to } v \in G\}|$. If the graph is a tree the reachability value r_i can be recursively defined as $1 + \sum_{j \in C(i)} r_j$, this is the **subtree size** of a node in a tree and when the context is a tree subtree size if often used interchangeably with reachability value.

Definition 3 *(Height, Depth, Level).* The **height** of a non-leaf node is one more than the maximum of the heights of its children and a leaf node's height is zero. The **depth** of a node is the number of edges in the path from that node to the root. The set of nodes at a particular depth forms the **level** at that depth.

Definition 4 *(Types of Trees).* A rooted tree with out-degree upper bounded by k is called a **k-ary tree** otherwise they are **general trees**. **Full trees** are k-ary trees where every out-degree is either k or 0. A **complete k-ary tree** is a k-ary tree with every level except possibly the last filled and all nodes in the last level filled from the left. The unique reachability sequence of such trees is denoted by $T_k^c(n)$. A degenerate tree is a tree which is just a path.

We now define the appropriate notions for the purpose of approximation.

Definition 5 *(Degree consistency).* We say that a graph G is **δ-in-degree consistent** with graph H if they have the same set of nodes and if for all nodes i the following holds: $I_H(i) \leq I_G(i) \leq \delta \cdot I_H(i)$. Here in-degree can be replaced with out-degree to get **δ-out-degree consistency**. If a graph G is both in-degree and out-degree consistent with graph H, then we say it is **δ-degree-consistent**.

Definition 6 *(Reachability consistency).* For **ρ-reachability consistency** we generalize the idea of reachability to get a similar notion of approximation as degree consistency. Given a tree we say that it is ρ-reachability consistent if for all nodes i the following holds: $r_i \leq 1 + \sum_{j \in C(i)} r_j \leq \rho \cdot r_i$, where r_i the given reachability value for node i in the sequence. The above notion of approximation can be extended to DAGs by replacing the inequality constraint with $r_i \leq O_G(i) + \max_{j \in C(i)} r_j \leq \rho \cdot r_i$.

Definition 7 *(Bicriteria Approximation).* We utilize the language of bicriteria optimization (see [38,39]) to say that G **(ρ, δ)-approximates** graph H if it is ρ-reachability consistent with the reachability sequence of H and it is δ-degree consistent with H. This captures the intuition that G approximately matches both the degree structure of G and its reachability sequence.

2.1. Synchronized and unsynchronized realization problems

We take attributes of nodes of DAGs, namely the degree, the reachability, the depth, and the height of a node and consider the problem of the existence of DAGs which has a sequence[4] of attributes that corresponds exactly to the given input. Most of our results consider two or more of these attributes for a node being available in a synchronized form, that is, the sequence is made of tuples consisting of attributes of a node. Sometimes we consider unsynchronized variants wherein we get multiple sequences one each for a particular attribute with no apriori synchronization or pairing of attributes to a particular corresponding node. More formally, these two variants problem variants can be defined as follows:

Definition 8 *(Synchronized sequence).* When a sequence of tuples made up of more than one attribute of the same node is given, then such a sequence is called a synchronized sequence. Refer to Example 2.

Example 2. Given a sequence of tuples of the form (reachability, depth): $\{(1,4),(2,3),(1,3),(1,3),(3,2),(3,2),(7,1),(1,1),(9,0)\}$, does there exist any DAG such that the sequence of such tuples is equal to the input sequence?

Definition 9 *(Unsynchronized sequences).* When multiple sequences are given, each corresponding to a different attribute of a node, then those sequences are called as unsynchronized sequences. Refer to Example 3.

Example 3. Consider sequences of reachability $\{1,2,3,1,1,3,7,1,9\}$ and that of depth $\{4,3,3,3,2,2,1,1,0\}$, does there exist a DAG such that the reachability sequence and the depth sequence are equal to the input sequences?

In almost all cases the realization problems being considered will be synchronized, especially when degrees of nodes are involved and this will be clear in context. When we do consider the unsynchronized variant especially where height and depth of nodes are involved, they will be explicitly referred to as unsynchronized realization problems.

3. Exact polynomial-time algorithms

We now discuss the particular realization problem variants and the sub-classes of DAGs which can be realized in polynomial time. These include some subclasses of trees where the reachability realization is polynomial time solvable (Section 3.1), a linear time algorithm for reachability realization in DAGs when there are no degree bounds (Section 3.2), an $O(n)$ algorithm for depth realization (Section 3.3) and an $O(n \log n)$ algorithm for height realization (Section 3.4).

3.1. Reachability realization in sub-classes of trees

1. *Complete Trees:* The complete tree on a given number of nodes n has a fixed structure. We just construct such a tree and then find the sequence. If on comparison they match, then we have a YES answer.
2. *Degenerate Trees:* For a sequence to correspond to that of a degenerate tree there should be exactly one instance of every number from 1 to n.
3. *Subtree Sizes synchronized with rank:* Suppose we are given the rank of the element as it would be in a binary search tree. This would allow trivial partitioning of the elements into either the left or the right subtree. That would divide the problem into two smaller problems. A divide and conquer method would then enable solving the problem in polynomial time. This variant sheds light on the fact that knowing how to partition is crucial in avoiding a combinatorial explosion, and together with the hardness reductions in Section 5 suggests the root cause of reachability realization to be many partitioning problems.

Note that these subclasses already come with specific degree restrictions. With minor modifications these algorithms suffice for realization problems on these variants when height and/or depth are present.

[4] Keeping with the convention of the graph realization problem, we use the term sequence instead of a multiset which is technically more apt.

3.2. Linear time algorithm for reachability realization in DAGs

We show that there exist polynomial-time verifiable, necessary and sufficient conditions that characterize reachability sequences of unbounded degree DAGs. This is reminiscent of conditions for the reconstruction of graphs given degree sequences (see [1]). However it is in contrast to the hardness result of degree realization for DAGs [11,13]. The inequalities in this section are deceptively simple considering the hardness results we prove in Section 5. Readers fond of puzzles are invited to prove the inequalities in Theorem 10 themselves before reading on.

Theorem 10 *(DAG reachability). Given a reachability sequence of natural numbers $\{r_1, r_2, \ldots, r_n\}$ in non-decreasing order the given sequence can be realized as a DAG iff $r_i \leq i$ for all i.*

Proof. In a DAG, nodes can only reach nodes with a strictly lower reachability otherwise its reachability would increase to a higher value causing a contradiction. Since there are at most $i-1$ nodes of lower reachability than r_i, it can reach at most i nodes including itself. Hence $r_i \leq i$ is a necessary condition. Next, for all i, connect i to the first $r_i - 1$ nodes. Excluding itself, i cannot reach more than $r_i - 1$ nodes since every node j it connects to can only connect to a node k with $k < j$ but i is already connected to k. Hence it can reach exactly r_i nodes and the inequality is a sufficient condition. □

3.3. Depth realization

Theorem 11 *(Depth Realization). Given a Depth sequence of natural numbers $\{D_1, D_2, \ldots, D_n\}$, the given sequence can be realized as a k-ary tree iff $count(d+1) \leq k \cdot count(d)$ for all d from 0 to $n-1$, where $count(i)$ is the frequency of i in the sequence.*

Proof. Depth values are the level at which a node is present. Hence, the frequency of a particular value will be the number of nodes present in a particular level. In a k-ary tree, every node can have at most k children, so level $d+1$ can accomodate at most $k \times count(d)$ nodes, where $count(i)$ is a function that returns the frequency of i in the sequence. Thus the algorithm simply checks whether $count(d+1) \leq k \cdot count(d)$ for every d from 0 to $n-1$. If all conditions hold, a tree can be constructed, otherwise no tree exists.

We need $O(n)$ time to find the frequency of all the depths and then check n conditions and hence the algorithm runs in $O(n)$ time where n is the number of elements in the input sequence. □

Slightly modified conditions exist for the following sub-classes:

- *Full k-ary tree:* Additionally check: $count(d) = k \times a$ where a is any positive integer, for all d.
- *Complete tree:* The condition is $count(d+1) = k \times count(d)$ for all d except possibly for the largest d.
- *Degenerate Tree:* Exactly one count of each value must be present.
- *General Tree:* $count(i) \geq 1$ for all i less than the maximum.
- *DAG with degree constraints:* For all d less than the maximum, $\sum_{i \text{ with depth}=d} O_G(i) = \sum_{j \text{ with depth}=d+1} I_G(j)$

3.4. Height realization

To solve height realization, we use the fact that a node can have height h only if one of its children has height $h-1$ implying that if a particular value exists in a given sequence then so would at least one instance of each value less than that. We define a *strand* of a node as a maximal path from the node to a leaf and use this notion to solve realization (see Fig. 2).

Theorem 12 *(Height Realization). Given a Height sequence of natural numbers $\{H_1, H_2, \ldots, H_n\}$, there is an $O(n)$ time algorithm to determine if the sequence can be realized as a tree.*

Proof. We use a greedy approach to divide the given sequence into maximal strands, by prioritizing strand length. So, first the root will get a strand of its own. Then, the next biggest remaining height value will get a strand and so on until no elements remain in the sequence.

This is a unique division because for a node to get a height value of h, it has to have at least one child whose height is $h-1$. Thus, a necessary condition for the existence of a tree is: if we get stuck at some point while making these strands then no tree can be constructed. Once these strands are constructed, we need to connect them together to get a tree, if possible. Any strand with its root's value as h can only be attached as a child of a node whose height value is at least $h+1$. A sufficient condition for a tree's existence is whether there are enough places where strands can be joined.

These conditions can be made into an algorithm as follows: Make an array of the count of each value in the sequence in descending order. Note that since all height values lie between 0 to n, we can use bucket sort to do this in $O(n)$ time and $O(n)$ space. Then compute $places = places + K \times count(i) - count(i-1)$ for all i. The *places* variable computes how many

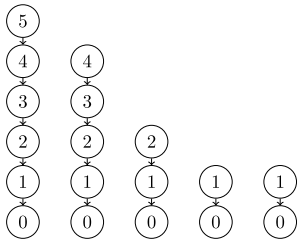

Fig. 2. Dividing the sequence {5, 4, 4, 3, 3, 2, 2, 2, 1, 1, 1, 1, 0, 0, 0, 0, 0} into strands.

places are left at higher levels after this strand has been added. For a tree to be possible, *places* must stay non-negative throughout, otherwise, the last strand we added was invalid and no tree would be possible. □

Simpler algorithms exist for the following sub-classes:

- *Complete K-ary:* Here we can just make the unique complete tree on n nodes, make corresponding sequences and compare the two sequences.
- *Full K-ary tree: places* must be zero at the end of the loop.
- *Degenerate Tree:* Exactly one count of each value must be present.
- *General Tree: count*$(i) \geq 1$ for all i less than the maximum.
- *DAG with degree constraints:* Additionally check that degree constraints are satisfied when creating and joining strands.

3.5. Realization of synchronized height and depth sequences

We can combine ideas from the methods to solve the height and depth problems to get an algorithm for solving the synchronized variant. Start by finding the strands ensuring that the depth values are assigned in order. Then put the roots of the strands at the correct depth. At every level check if a dedicated parent can exist for every strand that is rooted at that level. Do this in descending order of the root's height values. If one cannot find a parent at any point, no tree exists, otherwise, a tree exists.

4. Approximation algorithms for reachability realization

We present two approximation algorithms that are ρ-reachability consistent and δ-degree consistent with $\rho = \delta = O(\log n)$ given the reachability sequence along with the degree sequence. Thus the (ρ, δ)-approximation factor for our algorithms comes out to be $(O(\log n), O(\log n))$. The randomized algorithm runs in $O(n^{\omega+\max\{\frac{1}{18}, \omega-2, \frac{1-\alpha}{2}\}})$ time as detailed in Section 4.1 while the deterministic algorithm runs in $n^{O(k^3)}$ time as we detail in Section 4.2. Here, ω is the best known matrix multiplication exponent [40], α is its dual, and k is the maximum degree value. We compare further trade-offs between the two algorithms in Section 4.3 including the motivation for the more technically involved deterministic algorithm. While the exposition for both the algorithms addresses details using the full k-ary tree case, the results extend to all acyclic digraph cases in a straightforward manner by replacing reachability and degree consistency conditions for trees with that of DAGs.

4.1. LP based randomized rounding (LPRR) algorithm

The intuition behind the LPRR algorithm is to model the desired graph G as a collection of flows. Between every pair of nodes r_i and r_j with $r_i > r_j$ we assume a flow f_{ij} on each edge out of i and into j. We have three constraints for each node: the sum of flows into it is $I_G(i)$ (in-degree requirement), the sum of flows out of it is $O_G(i)$ (out-degree requirement), and that the reachability consistency conditions are satisfied. Finally, $f_{ij} = 0$ (no edge) for nodes i, j if node $r_i \leq r_j$.

$$\begin{aligned}
\min \quad & 1 \\
\text{s. t.} \quad & \sum_j f_{ji} = I_G(i) \ \ \forall i, && \text{In-degree requirement} \\
& \sum_j f_{ij} = O_G(i) \ \ \forall i, && \text{Out-degree requirement} \\
& r_i = 1 + \sum_j f_{ij} \cdot r_j \ \ \forall i, && \text{Reachability consistency} \\
& f_{ij} = 0 \ \ \forall i, j \text{ s.t. } r_i \leq r_j && \text{Acyclicity}
\end{aligned}$$

The LP is feasible as the graph G exists. After solving for a feasible set of f_{ij} values we round each edge ij to 1 with probability f_{ij} independently $24 \ln n$ times. Each time an edge is rounded to 1 it is added to the solution (initialized to a graph with all nodes in V but no edges). We use concentration bounds to show that the resulting structure satisfies the approximate reachability and degree consistency requirements with high probability.

In particular, the following multiplicative form of the Chernoff bound is used.

Theorem 13. *Let $X = \sum_{i=1}^{n} X_i$, where X_i are independent Bernoulli trials with $\Pr[X_i = 1] = p_i \; \forall \; 1 \leq i \leq n$ and let $\mu = \mathbb{E}[X] = \sum_{i=1}^{n} p_i$. For $\epsilon \in (0, 1)$*

$$\Pr[|X - \mu| \geq \epsilon] \leq 2e^{-\mu\epsilon^2/3} \tag{1}$$

Theorem 14 *(LPRR). Given a reachability sequence for a full k-ary tree, T, there exists a randomized $O(n^{\omega+\max\{\frac{1}{18}, \omega-2, \frac{1-\alpha}{2}\}})$-time algorithm that constructs a DAG that is an $(O(\log n), O(\log n))$-approximation to T.*

Proof. Analysis of running time: Clearly the bottleneck here is the LP solver and the state of the art solver runs in $O(n^{\omega+\max\{\frac{1}{18}, \omega-2, \frac{1-\alpha}{2}\}})$ time [26] where ω is the best known matrix multiplication exponent [40] and α is its dual. Further, under the common belief that $\omega \approx 2$ and $\alpha \approx 1$, our algorithm runs in $O(n^{\frac{37}{18}})$.

Proof of correctness: First, observe that vertex i has a total flow of $I_G(i)$ coming into it. So in one rounding the expected in-degree will be $I_G(i)$ and after $24 \ln n$ roundings the expected in-degree value will be $\mu_1 = 24 \ln n \cdot I_G(i)$. Invoking Chernoff bound with $\epsilon = 1/2$ we get that the probability that the node's in-degree lies outside the range $[\mu_1/2, 3\mu_1/2]$ is at most $2/n^2$. Similarly, we get the expected out-degree to be $\mu_2 = 24 \ln n \cdot O_G(i)$ and the probability that the out-degree of any node lies outside the range $[\mu_2/2, 3\mu_2/2]$ is at most $2/n^2$. Further for any node i, the reachability $1 + \sum_j f_{ij} \cdot r_j$ will have expected value $\mu_3 = 24 \ln n \cdot r_i$ and the probability that it lies outside the range $[\mu_3/2, 3\mu_3/2]$ is at most $2/n^2$. Applying the union bound over the $3n$ constraints in the LP, the probability that any of them lies outside their prescribed range is at most $3n \cdot \frac{2}{n^2} = o(1)$ as n goes to infinity. Thus with high probability after rounding, all of the quantities are within their prescribed ranges, i.e., the degree and reachability consistency are guaranteed to be within a logarithmic factor giving us the required (ρ, δ)-approximation. □

4.2. k-set packing based deterministic algorithm

We give the intuition behind the algorithm, DSHS (Deterministic Sieving using Hurkens-Schrijver) before presenting the technical details. DSHS runs in two (essentially independent) sieving phases, each phase taking $O(\log n)$ rounds: The MatchChildren phase matches each node (other than the leaves) with a (valid) set of children. The MatchParent phase matches each node (other than the root) with a parent.[5] Each phase starts with the entire set of candidate nodes and in each round sets up a $(k+1)$-set packing problem. The problem of k-set packing is to find the largest *disjoint* sub-collection of a given collection of sets each of cardinality k. The (approximate) solution to this problem sieves or reduces the candidate set by a constant factor, allowing each phase to finish in $O(\log n)$ rounds. Putting the results from the two phases together we get the desired (ρ, δ)-approximation factor. We use the following improvement of Hurkens-Schrijver's algorithm [28]. Note that a smaller ϵ improves the approximation but has a worse running time.

Theorem 15 *(Theorem 5, Furer-Yu [27]; with $\epsilon = \frac{1}{3}$). The (k+1)-Set Packing problem can be approximated to a factor $\frac{3}{k+2}$ in deterministic time $n^{O(k^3)}$.*

Theorem 16 *(DSHS). Given a reachability sequence $\{r_1, r_2, \ldots, r_n\}$ for a full k-ary tree, T, there exists a deterministic $n^{O(k^3)}$-time algorithm that constructs a DAG that is an $(O(\log n), O(\log n))$-approximation to T.*

Proof. DSHS (Deterministic Sieving using Hurkens-Schrijver): The method has two phases, MatchChildren ensures that every node has exactly k children that satisfy the reachability consistency condition and MatchParent ensures that every node has at least one parent. Both use set packing and we show in our proof that the specific set packing instances solved ensure the approximation notion in the theorem statement. Before the phases start, we initialize using an empty DAG with all the n nodes and no edges.

Phase MatchChildren: Initialize C_1 to be the set of all candidate nodes: nodes other than leaves (which have value 1). In round t the universe consists of C_t along with an entire set of V. Note that this has cardinality $|C_t| + |V|$ and is not the same as $C_t \cup V$. We create a collection of all possible $(k+1)$-sets with each set consisting of an element i from C_t and k elements, $j_1, j_2, \ldots, j_k$ from V such that $r_i = 1 + r_{j_1} + r_{j_2} + \ldots + r_{j_k}$. Note that each $(k+1)$-set is a possible match for i to its children. The existence of T guarantees that the optimal solution to this $(k+1)$-Set Packing problem has size $|C_t|$ – namely

[5] The root will have value n and leaves will have value 1.

the sub-collection consisting of each candidate node and its k children in T. Invoking the Hurkens-Schrijver approximation algorithm from the above theorem we are guaranteed to find a collection of sets that is at least $(3/(k+2)) \cdot |C_t|$. We use this sub-collection of sets to augment our solution DAG with the corresponding arcs from the node i to each of its children $(j_1, j_2, \ldots, j_k)$ for each set. We also remove the corresponding candidate nodes i from C_t to get C_{t+1}. Phase MatchChildren ends when the candidate set C_t is empty.

Phase MatchParent: Initialize P_1 to be the set of all candidate nodes: nodes other than leaves. In round t the universe consists of P_t along with an entire set of V. Note that this has cardinality $|P_t|+|V|$ and is not the same as $P_t \cup V$. We create a collection of all possible $(k+1)$-sets with each set consisting of one element, i from P_t and k elements from V, of which one, j is the parent and the remaining $k-1$ nodes $j_1, j_2, \ldots, j_{k-1}$ are siblings of i such that $r_j = 1 + r_i + r_{j_1} + r_{j_2} + \ldots + r_{j_{k-1}}$. Note that each $(k+1)$-set is a possible match for i to its parent j. The existence of T guarantees that the optimal solution to this $(k+1)$-Set Packing problem has size $|P_t|$ – namely the sub-collection consisting of each candidate node and its parent and siblings in T. Invoking the Hurkens-Schrijver approximation algorithm from the above theorem we are guaranteed to find a collection of sets that is at least $(3/(k+2)) \cdot |P_t|$. We use this sub-collection of sets to augment our solution DAG with the corresponding arcs from the node j to i and to each of its $k-1$ siblings for each set. We also remove the corresponding candidate nodes, i from P_t to get P_{t+1}. The Phase ends when the set P_t is empty.

Analysis of running time: The bottleneck step is the Hurkens-Schrijver approximation algorithm for set-packing which takes $n^{O(k^3)}$ time. Both of the phases take a logarithmic number of rounds, $\log_{\frac{k+2}{k-1}} n$ to be precise, which is absorbed into the total $n^{O(k^3)}$-time since the big-O is in the exponent.

Proof of correctness: Note that after phase MatchChildren every eligible node is matched to exactly k children satisfying the reachability consistency condition exactly. However, some nodes may not have parents and some may have too many parents. Still every node is guaranteed to get at most one parent per round and so no node has more than $O(\log n)$ parents at the end of Phase MatchChildren. Similarly, after phase MatchParent every eligible node has at least one parent. However some parents may get too many children. Yet, in each round a parent gets at most k children and so no node gets more than $O(k \log n)$ children. Thus at the end of the two phases we are guaranteed $O(\log n)$-degree consistency. Now observe that in each round of either phase, each node i either gets a valid set of k children, that is children $j_1, j_2, \ldots j_k$ such that $r_i = 1 + r_{j_1} + r_{j_2} + \ldots + r_{j_k}$, or no children at all; and we know that at the end of Phase MatchChildren every node other than leaves gets at least one valid set of children. Hence, we are guaranteed an $O(\log n)$-reachability consistent solution. Thus the solution DAG at the end of both phases is an (ρ, δ)-approximation to T. □

4.3. Trade-offs between the two approximation algorithms

The major trade-off between the DSHS and LPRR algorithms is the running time; while DSHS runs in $n^{O(k^3)}$, the LPRR algorithm is independent of k and runs in $O(n^\omega)$. Hence, unlike the deterministic algorithm, the randomized algorithm can be used even when k is a function of n. While these might suggest that the more complex and technically involved DSHS algorithm is inferior, that is not the case. The LPRR algorithm results in more complex solutions, in particular, LPRR may return digraphs with multi-edges while DSHS is guaranteed to return simple digraphs. Also, the multi-edges provide a tighter concentration of reachability consistency, albeit away from the reachability values, which may be a desirable property in applications where certainty is more important than consistency. While that is an important application, it is more common to require simple digraphs which the randomized algorithm cannot guarantee. This motivates the more technically involved DSHS algorithm.

5. Strong NP-completeness results

When the in-degrees and/or the out-degrees are constrained by the degree sequence we prove strong NP-completeness [41] using pseudo-polynomial transformations [42]. The reductions embed an instance of problems like 3-Partition between two consecutive levels of a tree. The most technically involved reduction proves the strong NP-completeness of the full k-ary tree realization problem (all in-degrees 1 or 0 and all out-degrees k or 0) in Section 5.3 which illustrates all the technicalities involved. This main reduction uses an intermediate problem called K-PwT which we prove is strongly NP-complete through a series of reductions in Section 5.1 and Section 5.2. We give a simpler reduction in Section 5.4 to prove that realization of general trees (no out-degree bound, all in-degrees 1 or 0) is also strongly NP-complete. In Section 5.5 we prove strong NP-completeness when there is only an out-degree bound (bounded out-degree DAGs).

5.1. NMTS-K *is strongly NP-complete*

We now prove the Strong NP-Completeness of the first of the two intermediate problems that are used in the reduction in Theorem 21.

Theorem 17 *(Due to Garey and Johnson [42]). The* NMTS *problem stated below is strongly NP complete:*

Given disjoint sets X and Y each containing m elements, a size function $s: X \cup Y \mapsto \mathbb{Z}^+$, and a target vector $B = (b_1, \ldots, b_m) \in \mathbb{N}^m$ with positive integer entries, can $X \cup Y$ be partitioned into m disjoint sets $A_1, A_2, \ldots, A_m$, each containing exactly one element from each of X and Y, such that, $\sum_{a \in A_i} s(a) = b_i$, for $1 \le i \le m$?

NMTS (X, Y, s, B, m) refers to an instance of the NMTS problem characterized by the sets X and Y, a size function s, the target vector B and the cardinality of the target vector m.

Theorem 18 *(*NMTS-K*). Given $K \geq 2$ disjoint sets X_i each containing m elements, a size function $s : \bigcup X_i \mapsto \mathbb{Z}^+$, and a target vector $B = (b_1, \ldots, b_m) \in \mathbb{N}^m$ with positive integer entries, it is strongly NP-complete to partition $\bigcup X_i$ into m disjoint sets $A_1, A_2, \ldots, A_m$, each containing exactly one element from each of X_i, such that, $\sum_{a \in A_i} s(a) = b_i$, for $1 \leq i \leq m$.*

NMTS-K (K, X_i, s, B, m) is an instance of the NMTS-K problem characterized by the integer K, the K sets X_i, a size function s, the target vector B and the cardinality of the target vector m.

Proof. The NMTS-K problem is in NP since given a candidate partition A_i, we only need to verify that, $\sum_{a \in A_i} s(a) = b_i$, for $1 \leq i \leq m$. We now construct an instance NMTS-K(K, X_i, s', B', m') of NMTS-K problem from an instance NMTS(X, Y, s, B, m) of the NMTS problem using the following transformation for $K \geq 3$ since for $K = 2$, the NMTS-K problem is the NMTS problem. Note that this is a polynomial transformation since computing Equations (3), (4) and (5) can be done in polynomial time.

$$m' = m,\ X_1 = X,\ X_2 = Y \tag{2}$$

$$X_i \text{ are disjoint sets such that } |X_i| = m' \text{ for } 3 \leq i \leq K \tag{3}$$

$$s'(x) = \begin{cases} s(x) & x \in X \cup Y \\ 1 & \text{otherwise} \end{cases} \tag{4}$$

$$B' = (b'_1, b'_2, \ldots, b'_m) \text{ where } b'_i = b_i + K - 2,\ \forall b_i \in B \tag{5}$$

We now prove that a YES instance of the NMTS-K problem occurs iff a YES instance of NMTS occurs. Every partition for the NMTS problem is associated with a partition for the NMTS-K problem. We denote the elements of X_i for $i \geq 3$ as $x_{ij}, 1 \leq j \leq m$ and let A_i be the partition for the NMTS problem. The associated partition for the NMTS-K problem A'_i, is defined as follows: $A'_i = A_i \cup \{x_{ji} | 3 \leq j \leq K\}$. This association immediately provides us with the equality: $\sum_{x \in A'_i} s'(x) = (\sum_{x \in A_i} s(x)) + (K - 2)$ which we compare with the relation $b'_i = b_i + (K - 2)$ from Eq. (5). We get that $\sum_{x \in A'_i} s'(x) = b'_i$ and $\sum_{x \in A_i} s(x) = b_i$ either happen simultaneously or not at all. Thus, this association ensures that this is a valid transformation.

The maximum number in the constructed instance is either the maximum size from X and Y or $K - 2$ added to the maximum number from B; all of which are polynomially bounded in the maximum integer in, and the length of, the NMTS instance, proving that NMTS-K is strongly NP-complete. □

5.2. K-PwT *is strongly NP-complete*

We now prove the Strong NP-Completeness of the second of the two intermediate problems that are used in the reduction in Theorem 21. The K-PwT problem can be regarded as a generalization of the 3-Partition problem where we are looking for a partition into K-sets and there are multiple targets to be reached instead of a single target.

Theorem 19 *(*K-PwT*). Given a set X with $|X| = Km$, $K \geq 2$, a size function $s : X \mapsto \mathbb{Z}^+$ and a target vector $B = (b_1, \ldots, b_m) \in \mathbb{N}^m$ with positive integer entries, it is strongly NP-complete to partition X into m disjoint sets $A_1, A_2, \ldots, A_m$, each containing exactly K elements, such that, $\sum_{a \in A_i} s(a) = b_i$, for $1 \leq i \leq m$.*

K-PwT (K, X, s, B, m) is an instance of the K-PwT problem characterized by the set X, an integer K, a size function s, the target vector B and the cardinality of the target vector m.

Proof. The K-PwT problem is in NP since given a particular candidate partition A_i, we only need to verify that, $\sum_{a \in A_i} s(a) = b_i$, for $1 \leq i \leq m$. We construct an instance K-PwT (K', X, s', B', m') of K-PwT problem from an instance NMTS-K (K, X_i, s, B, m) of the NMTS-K problem.

$$M = KmM' \text{ where, } M' = \max\left(\{s(x_i) | x_i \in X_i\} \cup \{b_i | b_i \in B\}\right) \tag{6}$$

$$K' = K,\ X = \bigcup X_i \tag{7}$$

$$\text{For } 1 \leq i \leq K \text{ and } \forall x_j \in X_i :\ s'(x_j) = s(x_j) + M^i \tag{8}$$

$$B' = (b'_1, b'_2, \ldots, b'_m) \text{ where } b'_j = b_j + \sigma,\ \forall b_j \in B \text{ and } \sigma = \sum_{i=1}^{K} M^i \tag{9}$$

M is polynomially bounded by the maximum integer in the NMTS-K instance. The transformation is polynomial since equations (6) to (9) are polynomial-time computable. Now we show that a YES instance of the K-PwT problem occurs iff a YES instance of NMTS-K occurs. For ease of exposition, for the rest of the proof, we write all the numbers in K-PwT (K', X, s', B', m') in base M. We make three remarks, the first: σ is a $K+1$ digit number with a 1 in all its digits except the rightmost or 0^{th} digit (Eq. (9)). The second, that every number $s'(x)$ has a 1 as its i^{th} digit, $s(x)$ in its rightmost digit[6] and 0 elsewhere. The third, a partition A_j for the NMTS-K instance $(\bigcup X_i)$ is also a partition for the K-PwT instance (X), irrespective of whether either of them solve the respective problems or not. We'll prove first that if A_j is a partition that solves the NMTS-K problem, then it also solves the K-PwT problem.

Let A_j be a partition that solves the NMTS-K problem. Using the same partition and Eq. (8) and (9), we get that $\sum_{x\in A_j} s'(x) = \sum_{x\in A_j} s(x) + \sum_{i=1}^{K} M^i = \sum_{x\in A_j} s(x) + \sigma = b_j + \sigma = b'_j$. This proves that if A_j solves the NMTS-K problem, then it also solves the K-PwT problem.

To prove the converse, let A_j be a partition that solves the K-PwT problem. We know that $\sum_{x\in A_j} s'(x) = b_j + \sigma$, which implies that $\sum_{x\in A_j} s(x) + \sum_i \sum_{x\in X_i\cap A_j} M^i = b_j + \sigma$ (from Eq. (8)). This in turn implies that $\sum_{x\in A_j} s(x) = b_j$ and $\sum_i \sum_{x\in X_i\cap A_j} M^i = \sigma = \sum_{i=1}^{K} M^i$ since $s(x)$ does not contribute to σ (from the first two remarks and Eq. (9)). Given $\sum_i \sum_{x\in X_i\cap A_j} M^i = \sum_{i=1}^{K} M^i$, equating the coefficients of the powers of M, we get that $|X_i \cap A_j| = 1, \forall i, j$ which says that every set in the partition contains exactly one element from each of the sets X_i. We already know from the earlier equations that $\sum_{x\in A_j} s(x) = b_j$. Thus, the partition A_j is a solution to the NMTS-K problem as well.

The maximum integer in the K-PwT instance created by the transformation, $\sigma + \max(b_1, \ldots, b_m)$, is bounded (from Eq. (6)) by a polynomial in the maximum integer in, and the length of, the NMTS instance, which by the definition of strong NP-completeness makes this is a pseudo-polynomial transformation. Thus, K-PwT is strongly NP-complete. □

5.3. Hardness of realization for full k-ary trees

We prove hardness of the realization problem for full k-ary trees by reduction from the K-PwT problem, which we show to be strongly NP-complete via an involved series of reductions in the full version. Since K-PwT is strongly NP-complete we can reduce from a subclass, Π_p, such that the largest number in the instance is polynomially bounded, formally, $\textsc{Max}[\mathrm{I}] \leq p(\textsc{Length}[\mathrm{I}]),\ \forall I \in \Pi_p$.

Problem 20 (K-PwT). Given a set X with $|X| = Km$, $K \geq 2$, sizes $s : X \mapsto \mathbb{Z}^+$ and a target vector $B = (b_1, \ldots, b_m) \in \mathbb{N}^m$, can X be partitioned into m disjoint sets $A_1, A_2, \ldots, A_m$, such that, $|A_i| = k$ and $\sum_{a\in A_i} s(a) = b_i$, for $1 \leq i \leq m$?

Theorem 21 *(Full k-ary tree). It is strongly NP-complete to determine whether a given reachability sequence can be realized as a full k-ary tree.*

Proof. The problem is clearly in NP since a tree acts as a certificate. Set $k = K$ and define a number M which is a power of K, is much greater in magnitude than any of the other numbers in the problem, and is polynomially bounded by the maximum integer in the K-PwT instance (Eq. (10)). We also define m' and m'' such that $m + m'$ and $m + m''$ are powers of K (Eq. (11)).

$$M_1 = \max\left(\{s(x_i)|x_i \in X\} \cup \{b_i|b_i \in B\}\right);\ M_2 = KmM_1;\ M = K^{\lceil \log_k M_2 \rceil} \tag{10}$$

$$m' = K^d - Km,\ m'' = K^{d-1} - m = m'/K, \text{where } d = \lceil \log_K(Km) \rceil \tag{11}$$

We make the sequence $S = C \cup P \cup G \cup D$ using four "component" sequences: the "child component" $C = C' \cup C''$, the "parent component" $P = P' \cup P''$, the "ancestor component" G and the "descendant component" D.

The "child component" C is the union of the C' and C'' while the "parent component" P is the union of the P' and P''. C' is in one-to-one correspondence with X, using the sizes of elements from X with M added to them while P' is in one-to-one-correspondence with B with changes to accommodate those made while making C'. The sets C'' and P'' ensure that the cardinality of C and P respectively are a power of K.

We construct the "ancestor component" in "levels". The lowest level l_{d-2} is constructed from P, by arbitrarily taking blocks of K elements, adding them all up and incrementing the result by one. Formally, order the elements in P arbitrarily as $P_1, P_2, \ldots, P_{K^{d-1}}$ and let $l_{d-2} = \{l_{d-2,i} \mid l_{d-2,i} = 1 + \sum_{j=1}^{K} P_{(i-1)K+j},\ 1 \leq i \leq K^{d-2}\}$. Other levels l_{d-i} are constructed in a similarly from levels l_{d-i+1}. This is continued until l_0 which has only one element since $|P|$ is a power of K and the size of each level above reduces by a factor of K. The element in l_0 would be the largest number in the final instance. The "descendant component" is constructed using reachability sequences of complete trees on the elements $c_i \in C$. For each

[6] Follows from Eq. (8) and M being greater than any number in the NMTS-K instance.

such c_i, we make a complete k-ary tree on c_i nodes and use its reachability sequence $T_k^c(c_i)$. The descendant component is then $D = \bigcup_i T_k^c(c_i)$. Since each $c_i \in C$ has the form $Kx+1$, $T_k^c(c_i)$ will also be full trees.

$$C' = \left\{K(s(x)+M)+1 \middle| x \in X\right\},\ C'' = \left\{\overbrace{KM+1, \ldots, KM+1}^{m' \text{times}}\right\} \tag{12}$$

$$P' = \left\{K(b_i+KM+1) \middle| b_i \in B\right\},\ P'' = \left\{\overbrace{K^2M+K, \ldots, K^2M+K)}^{m'' \text{times}}\right\} \tag{13}$$

$$G = \bigcup_{i=0}^{d-2} l_i, \text{ where "levels"} l_i \text{ are defined in text; } D = \bigcup_{i=1}^{K^d} T_k^c(c_i),\ \forall c_i \in C \tag{14}$$

Constructing S takes polynomial time as elements in P and C are derived directly from the K-PwT instance, there are a logarithmic number of levels each computed in polynomial time, and the polynomial number of trees in D each be computed in linear time.

By construction, elements in G and P will form a partial full k-ary tree with the elements in P as the "leaves", elements in C and D will make a forest with elements from C as roots of the trees in the forest, and C'' can be arbitrarily partitioned to connect to P'' elements. If there is a partition of X, we can partition C' accordingly to complete the tree.

To prove that a full k-ary tree implies a partition it is sufficient to prove that in any tree the set of children of P' is equal to C', that is, the nodes of P' and C' occur in consecutive levels in any tree. The node in l_0 is the largest and will necessarily have to be the root. This will be followed by the nodes from l_1 since no other nodes are large enough to reach those in l_0 (given the out-degree bound of k). Continuing the argument, $l_i \in G$ will always appear in consecutive levels in any tree and that P will follow below G. Since the in-degree is 1 no node from $p \in P$ will be a child of any $p' \in P$. Further, nodes in D will all be less than M/K in value and hence k of them will not be enough to reach nodes in P thus necessitating that all children of nodes of P come from C. We note that nodes from C' can not be children of nodes from P'' and so the set of children of P'' have to be C''. Since a value of the order of KM has to be reached for nodes in P' and all nodes in C' are of the order of M, all the nodes from C' will be used. Thus any tree will have nodes from P' and C' in consecutive levels and therefore have a partition. This proves NP-completeness and as the maximum integer used is polynomially bounded it also proves strong NP-completeness. □

5.4. Hardness of realization for general trees

Theorem 22 *(General Trees). It is strongly NP-complete to realize a general tree given a reachability sequence.*

Proof. This problem is in NP as a tree acts as a certificate and all that remains is give a reduction from 3-Partition.

Problem 23 *(3-Partition).* Given a set A, a target B and a size function $s : A \to Z^+$ such that $|A| = 3m$ and $B/4 < s(a_i) < B/2\ \forall a_i \in A$, can A be partitioned into m disjoint sets $A_1, A_2, \ldots, A_m$, such that for $1 \le i \le m$, $\sum_{a \in A_i} s(a) = B$?

Since this problem is strongly NP-complete [43], for the reduction, we use an instance of 3-Partition wherein the numbers are polynomially bounded in the input length. The notation $[n]$ is used for the set of the first n natural numbers. The constructed instance is:

$$S = \{m(B+1)+1\} \cup \{B+1, \ldots \ m \text{ times } \ldots, B+1\} \cup \left(\bigcup_{i=1}^{m} [s(a_i)]\right)$$

This construction is clearly polynomial time, all that remains is to show that this is valid reduction. In an abuse of notation the reachability size is often used to refer to the node with that reachability size.

In any potential tree, then the $m(B+1)+1$ node is forced to be the root as it cannot be the child of any node. Further, it will have m children, all the $B+1$ nodes as they cannot be the children of any other node. Considering the remaining nodes in a bottom up fashion, we see that a fixed structure is enforced on all these nodes due to the in-degree constraint in rooted trees. Any 2 node can only have a 1 as it's child, exhausting all 1's and leaving only 2's to be single children of 3's and only 3's as children for 4's and so on. By construction, there are exactly as many nodes of size $s-1$ as there are nodes of size s for all $s < B+1$ and hence they all get exhausted. This enforces that each node labeled $s(a_i)$ is a root of a path consisting of nodes with sub-tree sizes $[s(a_i)]$.

These restrictions are always present and a tree can be realized iff the paths rooted at $s(a_i)$ can be correctly made children of the $B+1$ nodes. This happens iff there is a partition of the 3-Partition instance without which the paths cannot be joined to the partial tree above it. □

5.5. Hardness of realization with out-degree constraints

Theorem 24 *(Bounded out-degree). It is strongly NP-complete to realize an acyclic graph given a reachability sequence and out-degree constraints.*

Proof. We reduce from 3-Partition, again using an instance with the maximum number polynomially bounded in the length of the problem. Let $s_i := s(a_i)$ and $M := mB^2$ and note that M is much bigger than every number in the 3-Partition instance and that the $\sum_i s_i = mB$. Let the reachability sequence $S := S_s \cup S_b \cup S_a$ where $S_s = \{Ms_i\}$, S_b be a multiset with m copies of $MB + 1$, and S_1 be a multiset of $mB - |A|$ copies of 1. Let the out-degree constraint for $MB + 1$ nodes be equal to three and for the s_i nodes be equal to s_i.

To achieve out-degree constraints, each of the s_i nodes needs to pick up $s_i - 1$ ones exhausting nodes in S_1. For the $MB + 1$ nodes to satisfy degree requirement they'll have to pick up exactly three nodes from S_s set which is possible iff there is a valid partition of the 3-Partition instance. □

5.6. Hardness corollaries of depth and height with reachability

Corollary 25. *Since depths were implicitly enforced in our reductions, a straightforward modification shows that both the synchronized and unsynchronized sequences of reachability and depths are NP-Complete.*

Corollary 26. *Both the synchronized and unsynchronized sequences of reachability and heights are NP-Complete.*

Proof. The same reduction for reachability can be used to prove this corollary by making implicit height values explicit. The reduction already ensures that nodes in the descendant component have a fixed structure. Combined with our definition of the big number M and the coefficients used this ensures that we know all height values in the tree. Since these can be determined via construction we simply explicitly use them in the reduction which would prove this corollary. □

Corollary 27 *(From Corollary 25 and Corollary 26). Synchronized and unsynchronized sequences of heights, depths and reachability are NP-Complete.*

6. Conclusion

In this paper we initiate the study of the realization problem for DAGs and rooted directed trees given reachability sequences, depth sequences and height sequences. We also consider synchronized or unsynchronized combination of these attributes, particularly with degree constraints added onto the reachability realization problem. We provide a linear time algorithm for DAGs with unbounded out-degrees and show hardness results for variants when we are also given a degree sequence bounding the in-degree and/or out-degree. We define a notion of bicriteria approximation based on reachability and degree consistency and give two $(O(\log n), O(\log n))$-approximation algorithms for all of these problems. For the depth and height realization problems, we gave algorithms and showed hardness results when combined with the reachability values.

Realization problems based on these global invariant properties are a rich area for study with many other interesting avenues. All of our results showcase the interplay between local node properties like degree and the global node properties like the reachability values and we believe that studying other properties can lead to new insights between various graph properties and structures. One future direction that is closely related to our work is understanding realization of in-reachability sequences (in-reachability value is the number of nodes that can reach a given node). Another direction for future work is the tightness of approximation of our current algorithms. It would be illuminating to either find algorithms with better approximation factors or come up with hardness of approximation results to prove tightness of approximation of the existing algorithms. A different direction that has not been considered until now is an active learning setting where we are allowed to ask queries on top of having reachability sequences, similar to the work in [44–46]. Such learning aspects are important in contexts where the reachability sequence is being updated dynamically.

Apart from these broader directions, the realization problems introduced in this work still have many directions for future research with many concrete questions that remain open. We conclude with some intriguing conjectures:

- *Conjecture: The LPRR algorithm can be derandomized using the method of conditional probability [47] in a manner that allows eliminating multi-edges in the algorithm's solutions.* This would allow combining the best features of the two approximation algorithms that we have presented.
- *Conjecture: The unsynchronized height and depth realization problem is strongly NP-complete.* Proving or disproving this conjecture can lead to a better understanding of how the depth and height of trees combine to inform the structure of the tree.
- *Conjecture: Given a uniform out-degree bound and a reachability sequence, the DAG realizability problem is solvable in polynomial time.* Our hardness result is for a non-uniform out-degree bound and we believe that the non-uniformity is necessary for hardness.

– *Conjecture: The general digraph reachability realization problem (with or without degree sequences) is strongly NP-complete.* All of our results were for acyclic graphs and we believe that addition of cycles will only make the problem harder.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We would like to thank Rahul Muthu for discussions about the subtree size reconstruction problem for trees.

References

[1] P. Erdos, T. Gallai, Graphs with points of prescribed degrees, Mat. Lapok 11 (1960) 132.
[2] S.L. Hakimi, On realizability of a set of integers as degrees of the vertices of a linear graph I, J. Soc. Ind. Appl. Math. 10 (1962) 496–506.
[3] V. Havel, A remark on the existence of finite graphs, Čas. Pěst. Mat. 80 (1955) 1253.
[4] M. Behzad, J.E. Simpson, Eccentric sequences and eccentric sets in graphs, Discrete Math. 16 (1976) 187–193.
[5] L. Lesniak, Eccentric sequences in graphs, Period. Math. Hung. 6 (1975) 287–293.
[6] A. Frank, Augmenting graphs to meet edge-connectivity requirements, SIAM J. Discrete Math. 5 (1992) 25–53.
[7] H. Frank, W. Chou, Connectivity considerations in the design of survivable networks, IEEE Trans. Circuit Theory 17 (1970) 486–490.
[8] M. Aigner, E. Triesch, Realizability and uniqueness in graphs, Discrete Math. 136 (1994) 3–20.
[9] A. Bar-Noy, K. Choudhary, D. Peleg, D. Rawitz, Realizability of graph specifications: characterizations and algorithms, in: International Colloquium on Structural Information and Communication Complexity, Springer, 2018, pp. 3–13.
[10] A. Berger, A note on the characterization of digraphic sequences, Discrete Math. 314 (2014) 38–41.
[11] A. Berger, M. Müller-Hannemann, How to attack the np-complete dag realization problem in practice, in: International Symposium on Experimental Algorithms, Springer, 2012, pp. 51–62.
[12] F. Harary, A survey of the reconstruction conjecture, in: Graphs and Combinatorics, Springer, 1974, pp. 18–28.
[13] S. Hartung, A. Nichterlein, Np-hardness and fixed-parameter tractability of realizing degree sequences with directed acyclic graphs, in: Conference on Computability in Europe, Springer, 2012, pp. 283–292.
[14] L. Lovász, A note on the line reconstruction problem, in: Classic Papers in Combinatorics, Springer, 2009, pp. 451–452.
[15] F. Harary, On the reconstruction of a graph from a collection of subgraphs, in: Theory of Graphs and Its Applications, Proc. Sympos. Smolenice, 1963, 1964, pp. 47–52.
[16] F. Harary, E. Palmer, On the problem of reconstructing a tournament from subtournaments, Monatshefte Math. 71 (1967) 14–23.
[17] S. Ulam, A Collection of Mathematical Problems, Wiley, 1960.
[18] P.J. Kelly, et al., A congruence theorem for trees, Pac. J. Math. 7 (1957) 961–968.
[19] C. Nash-Williams, Reconstruction of locally finite connected graphs with at least three infinite wings, J. Graph Theory 11 (1987) 497–505.
[20] A. Bar-Noy, K. Choudhary, D. Peleg, D. Rawitz, Graph profile realizations and applications to social networks, in: International Workshop on Algorithms and Computation, Springer, 2019, pp. 3–14.
[21] M. Mihail, N.K. Vishnoi, On generating graphs with prescribed vertex degrees for complex network modeling, in: Position Paper, Approx. and Randomized Algorithms for Communication Networks (ARACNE), Vol. 142, 2002.
[22] E. Mossel, N. Ross, Shotgun assembly of labeled graphs, IEEE Trans. Netw. Sci. Eng. 6 (2) (2017) 145–157.
[23] T. Akutsu, H. Nagamochi, Comparison and enumeration of chemical graphs, Comput. Struct. Biotechnol. J. 5 (2013) e201302004.
[24] S. Guindon, O. Gascuel, A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood, Syst. Biol. 52 (2003) 696–704.
[25] M.P. Ng, N.C. Wormald, Reconstruction of rooted trees from subtrees, Discrete Appl. Math. 69 (1996) 19–31.
[26] S. Jiang, Z. Song, O. Weinstein, H. Zhang, Faster dynamic matrix inverse for faster LPs, arXiv preprint, arXiv:2004.07470, 2020.
[27] M. Fürer, H. Yu, Approximating the k-set packing problem by local improvements, in: International Symposium on Combinatorial Optimization, Springer, 2014, pp. 408–420.
[28] C.A.J. Hurkens, A. Schrijver, On the size of systems of sets every t of which have an sdr, with an application to the worst-case ratio of heuristics for packing problems, SIAM J. Discrete Math. 2 (1989) 68–72.
[29] A. Berger, M. Müller-Hannemann, Dag realizations of directed degree sequences, in: International Symposium on Fundamentals of Computation Theory, Springer, 2011, pp. 264–275.
[30] B. Manvel, On reconstructing graphs from their sets of subgraphs, J. Comb. Theory, Ser. B 21 (1976) 156–165.
[31] B. Manvel, Reconstruction of trees, Can. J. Math. 22 (1970) 55–60.
[32] M. Steel, The complexity of reconstructing trees from qualitative characters and subtrees, J. Classif. 9 (1992) 91–116.
[33] B. Manvel, P.K. Stockmeyer, On reconstruction of matrices, Math. Mag. 44 (1971) 218–221.
[34] M. Dudík, L.J. Schulman, Reconstruction from subsequences, J. Comb. Theory, Ser. A 103 (2003) 337–348.
[35] J. Acharya, H. Das, O. Milenkovic, A. Orlitsky, S. Pan, String reconstruction from substring compositions, SIAM J. Discrete Math. 29 (2015) 1340–1371.
[36] D. Bartha, P. Burcsi, Reconstructibility of trees from subtree size frequencies, Stud. Univ. Babeş–Bolyai, Math. 59 (2014).
[37] W.J. Bruno, N.D. Socci, A.L. Halpern, Weighted neighbor joining: a likelihood-based approach to distance-based phylogeny reconstruction, Mol. Biol. Evol. 17 (2000) 189–197.
[38] M. Ehrgott, X. Gandibleux, A survey and annotated bibliography of multiobjective combinatorial optimization, OR Spektrum 22 (2000) 425–460.
[39] M.V. Marathe, R. Ravi, R. Sundaram, S. Ravi, D.J. Rosenkrantz, H.B. Hunt III, Bicriteria network design problems, J. Algorithms 28 (1998) 142–171.
[40] F. Le Gall, Powers of tensors and fast matrix multiplication, in: Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation, ACM, 2014, pp. 296–303.
[41] M.R. Garey, D.S. Johnson, "strong" NP-completeness results: motivation, examples, and implications, J. ACM 25 (1978) 499–508.
[42] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, WH Free. Co., San Fr, 1979.
[43] M.R. Garey, D.S. Johnson, Complexity results for multiprocessor scheduling under resource constraints, SIAM J. Comput. 4 (1975) 397–411.
[44] C. Daskalakis, R.M. Karp, E. Mossel, S.J. Riesenfeld, E. Verbin, Sorting and selection in posets, SIAM J. Comput. 40 (2011) 597–622.
[45] E. Emamjomeh-Zadeh, D. Kempe, Adaptive hierarchical clustering using ordinal queries, in: Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics, 2018, pp. 415–429.
[46] Z. Wang, J. Honorio, Reconstructing a bounded-degree directed tree using path queries, CoRR, arXiv:1606.05183 [abs], 2016. URL: http://arxiv.org/abs/1606.05183, arXiv:1606.05183.
[47] N. Alon, J.H. Spencer, The Probabilistic Method, 4th ed., Wiley Publishing, 2016.