Realization Problems on Reachability Sequences

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

Northeastern University, Boston, USA
mdippel@ccs.neu.edu, r.sundaram@northeastern.edu, akshar@ccs.neu.edu

Abstract. The classical Erdös-Gallai theorem 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 both a local constraint via degree sequences and a global constraint via a sequence of reachability values (number of nodes reachable from a given node). 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 two conjectures that we hope motivate further study of realizability with reachability constraints.

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

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 [12] on degree sequences [20,23] many other properties have been considered in the literature ranging from eccentricities [7,26] to connectivity and flow [13,14]. The best studied among these remain extensions of realization given degree sequences [1,5] and variants focusing on different subclasses of graphs [8,21,27]. In addition to their theoretical significance, realization questions occur naturally in numerous application contexts, including network design [14], social networks [6,29], DNA sequencing [30], enumerating chemical compounds [2], and phylogeny and evolutionary tree reconstruction [19,31].

We consider the realization problem on digraphs: we are given as input a sequence of tuples $(r_i, \mathit{I}(i), \mathit{O}(i))$, where r_i is the reachability value[1], $\mathit{I}(i)$ the in-degree and $\mathit{O}(i)$ the out-degree of each node $i, 1 \leq i \leq n$; 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

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

captured by the reachability sequence. Like all realization problems this has connections to the graph isomorphism problem and graph canonization.

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 goal may be to construct resilient networks that restrict the spread of the virus.

1.1 Our Contributions

We characterize the complexity of the realizability of acyclic digraphs as summarized in Table1. Instead of referring to bounded vs. unbounded in-degree, we simply talk about trees vs. DAGs since the in-degree is bounded by one in trees and they capture the essential behavior of bounded in-degree digraphs. This usage makes the exposition of the hardness results more natural.

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

- In Section 3 we give linear time verifiable, necessary and sufficient conditions, for realizing unbounded out-degree DAGs (Theorem 1).
- We define a notion of bicriteria approximation in Section 2 and give two algorithms in Section 4 to solve the reachability realization problem, both achieving an $(O(\log n), O(\log n))$-approximation.
  - Theorem 3: Randomized algorithm using LP rounding that runs in $O(n^\omega)$ time where ω is the matrix multiplication exponent.
  - Theorem 5: Deterministic algorithm using k-set packing heuristics [15,24] 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 6). We also give simpler reductions from 3-Partition for reachability realization when only one of in-degree or out-degree is bounded (Theorems 7, 8).

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 8, prove that reachability realization problems are strongly NP-complete even when the degree bounds are uniform. In particular, we note that the Theorems 6 and 7 which have uniform degree bounds rely on having the in-degree bounded while in Theorem 8, where only the out-degree is bounded, we are only able to show hardness in the non-uniform case.

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 $\mathsf{C}(i)$ denote the set of children of i, i.e., $\mathsf{C}(i) = \{j|(i,j) \in G\}$. The **out-degree** of i, $\mathsf{O}_G(i)$ is the number of its children, i.e., $|\mathsf{C}(i)|$; the **in-degree** of node i in graph G, $\mathsf{I}_G(i)$ is the number of nodes with arcs directed into i. 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 $1 + \sum_{j\in \mathsf{C}(i)} r_j$. 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 $T^c_k(n)$.

We now define the appropriate notions for the purpose of approximation. 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: $\mathsf{I}_H(i) \le \mathsf{I}_G(i) \le \delta \cdot \mathsf{I}_H(i)$. 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**. 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: $a_i \le 1 + \sum_{j\in \mathsf{C}(i)} a_j \le \rho \cdot a_i$, where a_i is the reachability label on node i in the approximate solution. The above notion of approximation can be extended to DAGs by replacing the inequality constraint with $a_i \le \mathsf{O}_G(i) + \max_{j\in \mathsf{C}(i)} a_j \le \rho \cdot a_i$. Finally, we utilize the language of bicriteria optimization (see [11,28]) 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 structure of G and its reachability sequence.

3 Linear time algorithm for DAGs

We show that there exist polynomial-time verifiable, necessary and sufficient conditions that characterize reachability sequences of unbounded out-degree DAGs. This is reminiscent of conditions for the reconstruction of graphs given degree sequences (see [12]). However it is in contrast to the hardness result of degree realization for DAGs [9,22]. 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 1 themselves before reading on.

Theorem 1 (DAG reachability). *Given a sequence of natural numbers* $\{r_1, r_2, \ldots, r_n\}$ *in non-decreasing order there exists a DAG for which the given sequence is the sequence of the reachability sizes of the 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. Observe that, 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 also a sufficient condition. □

4 Approximation Algorithms

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 $O(n^{\omega})$ 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 . 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.

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 $\mathsf{I}(i)$ (in-degree requirement), the sum of flows out of it is $\mathsf{O}_G(i)$ (out-degree requirement), and that the reachability consistency conditions are satisfied. Further, there cannot be an edge (f_{ij} must be 0) from node i to node j if node i has a smaller reachability value than node j.

The existence of G guarantees that the LP is feasible. 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 then argue that the resulting structure satisfies the approximate reachability and degree consistency requirements with high probability using concentration bounds.

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

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

Theorem 2. *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* $\mu = \mathbb{E}[X] = \sum_{i=1}^{n} p_i$. *Then, for* $0 < \epsilon < 1$

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

Theorem 3 (LPRR). *Given a reachability sequence for a full k-ary tree, T, there exists a randomized $O(n^\omega)$-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 LP solver is the bottleneck in this algorithm and the state of the art solver runs in $O(n^\omega)$ time [10] where ω is the current best known exponent for matrix multiplication [25].
Proof of correctness: First, observe that vertex i has a total flow of $I(i)$ coming into it. So in one rounding the expected in-degree will be $I(i)$ and after $24 \ln n$ roundings the expected in-degree value will be $\mu_1 = 24 \ln n \cdot I(i)$. 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.[2] 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 [24]. Note that a smaller ϵ can improve the approximation but comes at the cost of a worse running time.

Theorem 4 (Theorem 5, Furer-Yu [15]; with $\epsilon = \frac{1}{3}$). *The (k+1)-Set Packing problem can be approximated to a factor $\frac{3}{k+3}$ in deterministic time $n^{O(k^3)}$.*

Theorem 5 (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):** 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 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 becomes 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

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

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}. Phase MatchParent ends when the candidate set P_t becomes empty.

Analysis of running time: The bottleneck step is running the Hurkens-Schrijver approximation algorithm for set-packing which takes $n^{O(k^3)}$ time. Each of the two phases takes 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. $\square$

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. We also note that LPRR can be derandomized using the method of conditional probability [3]. 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 [17] using pseudo-polynomial transformations [18]. The reductions embed an instance of problems like 3-Partition between two consecutive levels of a tree. We first present the proof for the full k-ary tree realization problem (all in-degrees 1 or 0 and all out-degrees k or 0) in Section 5.1 which illustrates all the technicalities involved. We give a simpler reduction (in Section 5.2) to prove that realization of general trees (no out-degree bound) is also strongly NP-complete. In Section 5.3 we give a reduction to prove strong NP-completeness when there is only an out-degree bound.

5.1 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 prove to be strongly NP-complete in the appendix via a series of involved reductions. 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 1 (*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 6 (Full k-ary tree). *It is strongly NP-complete to determine the existence of a full k-ary tree whose reachability sequence equals a given sequence.*

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. 2). We also define m' and m'' such that $m + m'$ and $m + m''$ are powers of K (Eq. 3).

$$M_1 = \max\left(\{s(x_i) | x_i \in X\} \cup \{b_i | b_i \in B\}\right); M_2 = KmM_1; M = K^{\lceil \log_k M_2 \rceil} \quad (2)$$

$$m' = K^d - Km,\ m'' = K^{d-1} - m = m'/K, \text{ where } d = \lceil \log_K(Km) \rceil \quad (3)$$

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 the set 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 to sizes of elements of X 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 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.

$$C' = \big\{K(s(x)+M)+1 \big| \, x \in X\big\},\ C'' = \big\{\overbrace{KM+1, \ldots, KM+1}^{m' \text{times}}\big\} \tag{4}$$

$$P' = \big\{K(b_i+KM+1) \big| \, b_i \in B\big\},\ P'' = \big\{\overbrace{K^2M+K, \ldots, K^2M+K)}^{m'' \text{times}}\big\} \tag{5}$$

$$G = \bigcup_{i=0}^{d-2} l_i,\ \text{where "levels"} l_i \text{ are defined in text; } D = \bigcup_{i=1}^{K^d} T_k^c(c_i),\ \forall c_i \in C \tag{6}$$

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. All that remains is to prove that the reduction is valid.

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. □

5.2 Hardness of Realization for General Trees

Theorem 7 (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 2 (*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 $A_1, A_2, \dots, A_m$, such that for $1 \leq i \leq m$, $\sum_{a \in A_i} s(a) = B$?

This problem is strongly NP-complete [16] which implies that even if the numbers in the instance are assumed to be polynomially bounded in the input length, the problem is still NP-complete. We use such an instance of 3-PARTITION to create a reachability sequence as follows. The notation $[n]$ is used for the set consisting of the first n natural numbers. The constructed instance is as follows:

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

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 labelled $s(a_i)$ 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; if there is no partition then the paths cannot be joined to the partial tree above it to form a single tree. □

5.3 Hardness of Realization with out-degree constraints

Theorem 8 (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 $S \coloneqq S_s \cup S_b \cup S_a$ 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 will need to pick up $s_i - 1$ ones exhausting nodes in S_1. For the $MB+1$ nodes to achieve their degree requirement they'll have to pick up exactly three nodes from S_s set which will be possible iff there is a valid partition of the 3-Partition instance. □

6 Conclusion

In this paper we initiate the study of the realization problem for DAGs and rooted directed trees given a reachability sequence. 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. We conclude with two intriguing conjectures:

- Given a uniform out-degree bound and a reachability sequence the DAG realizability problem is solvable in poly-time.
- The general digraph realizability problem given a reachability sequence (with or without degree sequences) is strongly NP-complete.

References

1. Aigner, M., Triesch, E.: Realizability and uniqueness in graphs. Discrete Mathematics **136**(1-3), 3–20 (1994)
2. Akutsu, T., Nagamochi, H.: Comparison and enumeration of chemical graphs. Computational and structural biotechnology journal **5**(6), e201302004 (2013)
3. Alon, N., Spencer, J.H.: The Probabilistic Method. Wiley Publishing, 4th edn. (2016)
4. Angluin, D., Valiant, L.G.: Fast probabilistic algorithms for hamiltonian circuits and matchings. Journal of Computer and system Sciences **18**(2), 155–193 (1979)
5. Bar-Noy, A., Choudhary, K., Peleg, D., Rawitz, D.: Realizability of graph specifications: Characterizations and algorithms. In: International Colloquium on Structural Information and Communication Complexity. pp. 3–13. Springer (2018)
6. Bar-Noy, A., Choudhary, K., Peleg, D., Rawitz, D.: Graph profile realizations and applications to social networks. In: International Workshop on Algorithms and Computation. pp. 3–14. Springer (2019)
7. Behzad, M., Simpson, J.E.: Eccentric sequences and eccentric sets in graphs. Discrete Mathematics **16**(3), 187–193 (1976)
8. Berger, A.: A note on the characterization of digraphic sequences. Discrete Mathematics **314**, 38–41 (2014)
9. Berger, A., Müller-Hannemann, M.: How to attack the np-complete dag realization problem in practice. In: International Symposium on Experimental Algorithms. pp. 51–62. Springer (2012)

10. Cohen, M.B., Lee, Y.T., Song, Z.: Solving linear programs in the current matrix multiplication time. In: Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing. pp. 938–942. ACM (2019)
11. Ehrgott, M., Gandibleux, X.: A survey and annotated bibliography of multiobjective combinatorial optimization. OR-Spektrum **22**(4), 425–460 (2000)
12. Erdos, P., Gallai, T.: Graphs with points of prescribed degrees. Mat. Lapok **11**(264-274), 132 (1960)
13. Frank, A.: Augmenting graphs to meet edge-connectivity requirements. SIAM Journal on Discrete Mathematics **5**(1), 25–53 (1992)
14. Frank, H., Chou, W.: Connectivity considerations in the design of survivable networks. IEEE Transactions on Circuit Theory **17**(4), 486–490 (1970)
15. Fürer, M., Yu, H.: Approximating the k-set packing problem by local improvements. In: International Symposium on Combinatorial Optimization. pp. 408–420. Springer (2014)
16. Garey, M.R., Johnson, D.S.: Complexity results for multiprocessor scheduling under resource constraints. SIAM Journal on Computing **4**(4), 397–411 (1975)
17. Garey, M.R., Johnson, D.S.: "Strong"NP-Completeness Results: Motivation, Examples, and Implications. Journal of the ACM (JACM) **25**(3), 499–508 (1978)
18. Garey, M.R., Johnson, D.S.: Computers and Intractability: A guide to the theory of NP-completeness. WH Free. Co., San Fr (1979)
19. Guindon, S., Gascuel, O.: A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. Systematic biology **52**(5), 696–704 (2003)
20. Hakimi, S.L.: On realizability of a set of integers as degrees of the vertices of a linear graph I. Journal of the Society for Industrial and Applied Mathematics **10**(3), 496–506 (1962)
21. Harary, F.: A survey of the reconstruction conjecture. In: Graphs and Combinatorics, pp. 18–28. Springer (1974)
22. Hartung, S., Nichterlein, A.: Np-hardness and fixed-parameter tractability of realizing degree sequences with directed acyclic graphs. In: Conference on Computability in Europe. pp. 283–292. Springer (2012)
23. Havel, V.: A remark on the existence of finite graphs. Casopis Pest. Mat **80**(477-480), 1253 (1955)
24. Hurkens, C.A.J., Schrijver, A.: 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 Journal on Discrete Mathematics **2**(1), 68–72 (1989)
25. Le Gall, F.: Powers of tensors and fast matrix multiplication. In: Proceedings of the 39th international symposium on symbolic and algebraic computation. pp. 296–303. ACM (2014)
26. Lesniak, L.: Eccentric sequences in graphs. Periodica Mathematica Hungarica **6**(4), 287–293 (1975)
27. Lovász, L.: A note on the line reconstruction problem. In: Classic Papers in Combinatorics, pp. 451–452. Springer (2009)
28. Marathe, M.V., Ravi, R., Sundaram, R., Ravi, S., Rosenkrantz, D.J., Hunt III, H.B.: Bicriteria network design problems. Journal of algorithms **28**(1), 142–171 (1998)
29. Mihail, M., Vishnoi, N.K.: On generating graphs with prescribed vertex degrees for complex network modeling. Position Paper, Approx. and Randomized Algorithms for Communication Networks (ARACNE) **142** (2002)
30. Mossel, E., Ross, N.: Shotgun assembly of labeled graphs. IEEE Transactions on Network Science and Engineering (2017)

31. Ng, M.P., Wormald, N.C.: Reconstruction of rooted trees from subtrees. Discrete Applied Mathematics **69**(1-2), 19–31 (1996)

A Proofs of strong NP-completeness for intermediate problems

We now prove Strong NP-Completeness of problems used to prove Theorem 6.

Theorem 9 (Due to Garey and Johnson [18]). *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 $B = (b_1, \dots, b_m)\ \in \mathbb{N}^m$ with positive integer entries, can $X \cup Y$ be partitioned into m disjoint sets $A_1, A_2, \dots, A_m$, each containing exactly one element from each of X and Y, such that, $\sum_{a \in A_i} s(a) = b_i$, for $1 \leq i \leq m$?

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.

A.1 NMTS-K is strongly NP-complete

The NMTS-K problem is proved strongly NP-complete by reduction from NMTS.

Theorem 10 (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 $B = (b_1, \dots, b_m)\ \in \mathbb{N}^m$ with positive integer entries, it is strongly NP-complete to partition $\bigcup X_i$ into m disjoint sets $A_1, A_2, \dots, A_m$, 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 8, 9 and 10 can be done in polynomial time.

$$m' = m,\ X_1 = X,\ X_2 = Y \tag{7}$$

$$X_i \text{ are disjoint sets such that } |X_i| = m' \text{ for } 3 \leq i \leq K \tag{8}$$

$$s'(x) = \begin{cases} s(x) & x \in X \cup Y \\ 1 & \text{otherwise} \end{cases} \tag{9}$$

$$B' = (b'_1, b'_2, \dots, b'_m) \text{ where } b'_i = b_i + K - 2,\ \forall b_i \in B \tag{10}$$

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. 10. 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 the target B; all of which are polynomially bounded in the maximum integer in, and the length of, the NMTS instance. This proves that NMTS-K is strongly NP-complete.

A.2 K-PwT is strongly NP-complete

The K-PwT problem is strongly NP-complete by reduction from the NMTS-K problem. This 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 11 (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 in the following manner. M is polynomially bounded by the maximum integer in the NMTS-K instance.

$$M = KmM' \text{ where, } M' = \max\left(\{s(x_i) | x_i \in X_i\} \cup \{b_i | b_i \in B\}\right) \tag{11}$$

$$K' = K, \; X = \bigcup X_i \tag{12}$$

$$\text{For } 1 \leq i \leq K \text{ and } \forall x_j \in X_i : \; s'(x_j) = s(x_j) + M^i \tag{13}$$

$$B' = (b'_1, b'_2, \ldots, b'_m) \text{ where } b'_j = b_j + \sigma, \; \forall b_j \in B \text{ and } \sigma = \sum_{i=1}^{K} M^i \tag{14}$$

The transformation is polynomial since equations 11 to 14 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. 14). The second, that every number $s'(x)$ has a 1 as its i^{th} digit, $s(x)$ in its rightmost digit[3] 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. 13 and 14, 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. 13). 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. 14). 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. 11) 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.

[3] Follows from Eq. 13 and M being greater than any number in the NMTS-K instance.