Discrete Applied Mathematics 332 (2023) 47–61

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

Constructing bounded degree graphs with prescribed degree and neighbor degree sequences

Uroš Čibej [a], Aaron Li [b], István Miklós [c,d,*], Sohaib Nasir [e], Varun Srikanth [f]

[a] University of Ljubljana, Faculty of Computer and Information Science, Večna pot 113, 1000 Ljubljana, Slovenia
[b] University of Minnesota, 206 Church St. SE, Minneapolis, MN, 55455, USA
[c] Rényi Institute, ELKH, 1053 Budapest, Reáltanoda u. 13-15, Hungary
[d] SZTAKI, ELKH, 1111 Budapest, Lágymányosi u. 11, Hungary
[e] Vassar College, 124 Raymond Avenue, Poughkeepsie, NY 12604, USA
[f] Georgia Institute of Technology North Avenue, Atlanta, GA 30332, USA

ARTICLE INFO

Article history:
Received 30 March 2022
Received in revised form 16 January 2023
Accepted 1 February 2023
Available online 8 February 2023

Keywords:
Degree sequences
Neighbor degree constraint
NMR spectroscopy

ABSTRACT

Let $D = (d_1, d_2, \ldots, d_n)$ and $F = (f_1, f_2, \ldots, f_n)$ be two sequences of positive integers. We consider the following decision problems: is there a (i) multigraph, (ii) loopless multigraph, (iii) simple graph, (iv) cycle-free graph (forest or tree), (v) caterpillar $G = (V, E)$ such that for all k, $d(v_k) = d_k$ and $\sum_{w\in\mathcal{N}(v_k)} d(w) = f_k$ ($d(v)$ is the degree of v and $\mathcal{N}(v)$ is the set of neighbors of v). Here we show that all these decision problems can be solved in polynomial time if $\Delta := \max_k d_k$ is bounded. The problems are converted into an integer programming feasibility problem in which both the number of variables and the number of inequalities depend only on Δ but not on n.

The problem is motivated by NMR spectroscopy of hydrocarbons. The algorithm has been implemented in the ZIMPL language, and its applicability is demonstrated on trees up to $n = 1000$ vertices. The average reconstruction time for trees with 1000 vertices is still less than 40 ms.

1. Introduction

In statistical testing of networks, a network from real life must be compared with a background distribution of random graphs. In such null models of random graphs, the degree sequence of the graphs is typically fixed. However, there are structural properties that are not preserved by the degree sequence. One example for such a property is the *assortativity* [17]. A network is called *assortative* if vertices tend to be connected to vertices with similar degrees, and it is called *dissortative* if low degree vertices tend to connect to high degree vertices. The observation that real life networks with similar degree sequences might have different assortativity urged research to develop null models that preserve structural properties above the degree sequence. Such models are, for example, the joint degree matrix model [4,17] that prescribes the number of edges between vertex classes with given degrees and the *dk-random graphs* that considers the distribution of induced sub-graphs up to a given size [15]. While there are polynomial algorithms to construct a realization of a joint degree matrix [4], constructing a graph with prescribed small sub-graph distribution appears to be a hard problem. This motivated the question: "What kind of local properties make the graph construction hard?" Erdős and Miklós showed that it is already NP-complete to ask if there exists a graph with given degree and neighbor degree sequence [7].

* Corresponding author at: Rényi Institute, ELKH, 1053 Budapest, Reáltanoda u. 13-15, Hungary.
E-mail address: miklos.istvan@renyi.hu (I. Miklós).

https://doi.org/10.1016/j.dam.2023.02.004

In this paper, we show that there exists a polynomial algorithm to construct graphs with prescribed degree and neighbor degree sequence if there is a bound on the maximum degree. We consider several variants of the problem: we might require that the graph be a simple graph, cycle-free graph, or caterpillar. Particular emphases are on the cases when the graph is a tree as it also has an application of obtaining the chemical structure of hydrocarbons by NMR spectroscopy data.

Hydrocarbons are the simplest organic molecules consisting of only hydrogen and carbon atoms. A carbon atom makes four covalent bonds, while a hydrogen atom makes only one covalent bond. What follows is that hydrogen atoms can bond only to carbon atoms in hydrocarbons. Therefore, the information on the covalent bonds between carbon atoms completely describes the chemical structure of a hydrocarbon molecule. Indeed, if a carbon atom makes k covalent bonds to other carbon atoms, then it must make $4 - k$ bonds to hydrogen atoms. The diagram representing only the covalent bonds between the carbon atoms is a connected graph with maximum degree 4. This graph is called the skeletal structure and has complete information about the chemical structure. A hydrocarbon might contain multiple bonds between two carbon atoms. Such a hydrocarbon is called unsaturated, and the skeletal structure is a connected, loopless multigraph in that case. When there are only single bonds between carbon atoms, the hydrocarbon is called saturated, and then the skeletal structure is a connected simple graph.

The chemical formula C_nH_m means that there are n carbon atoms and m hydrogen atoms in a molecule. There are simple chemical measurements to obtain n and m. When $m = 2n + 2$, the skeletal structure is a tree. Indeed, in that case, there are $n - 1$ covalent bonds between carbon atoms, and any connected graph with n vertices and $n - 1$ edges is a tree. As m decreases, the number of covalent bonds between carbon atoms increases. This might be obtained by multiple bonds between carbon atoms (that is, we are talking about unsaturated hydrocarbons) or making cycles in the skeletal structure. Unsaturated hydrocarbons react with halogens while saturated hydrocarbons do not react, and that simple chemical reaction helps separate unsaturated hydrocarbons from saturated hydrocarbons with cycles in their skeletal structure. In conclusion, the chemical formula C_nH_m is easy to obtain and it is easy to decide if the hydrocarbon is saturated or not as well.

More information on the chemical structure of a hydrocarbon can be attained from NMR spectroscopy. Roughly speaking, the position (frequency) of a peak in the NMR spectroscopy depends on the number of hydrogen atoms bonding to a particular carbon atom causing the peak, and the size of the peak tells the number of such carbon atoms. Furthermore, a peak (roughly, a Gaussian bell-shaped curve) is split depending on the number of hydrogen atoms bonding to neighbor carbon atoms. A peak is split into $l + 1$ parts when there are altogether l hydrogen atoms on the neighbor carbon atoms. From this information, the degree of a carbon atom as well as the sum of the degrees of the neighbor carbon atoms in the skeletal structure can be obtained. Indeed, if a peak related to k hydrogen–carbon bonds is split into $l + 1$ parts, then the degree of the carbon atom in the skeletal structure is $4 - k$, and the sum of the degrees of the neighbor carbon atoms is $4 \times (4 - k) - l$.

This raises the following graph theoretical question: given degree sequences $D = (d_1, d_2, \dots, d_n)$ and $F = (f_1, f_2, \dots, f_n)$, is there a graph $G = (V, E)$, such that for all k, $d(v_k) = d_k$ and $\sum_{w \in \mathcal{N}(v_k)} d(w) = f_k$, where $d(v)$ is the degree of vertex v and $\mathcal{N}(v)$ is the set of the neighbors of v? Erdős and Miklós showed that this decision problem is NP-complete in general [7]. Here we consider a specific case: the maximum degree is 4, and we are looking for tree realizations. We show that this specific case can be solved in polynomial time.

There is typically more than one graph with a given degree and neighbor degree sequence. Some of the solutions are chemically not stable due to spherical constraints. For example, it is known that the hydrocarbon whose skeletal structure would be the balanced unrooted ternary tree with $17(= 1 + 4 + 12)$ vertices is chemically not stable. In principle, molecules that have skeletal structure with larger diameter tend to be more stable. Among trees with given degree sequence, caterpillars have the maximum diameter. This motivates us to ask when a caterpillar with given degree and neighbor degree sequence exists.

The paper is structured as follows. In the Preliminaries, we give some definitions on graphs and also introduce the concept of labeled stub-stars. When each edge in a graph is cut into two half-edges (stubs), we get stub-stars. If each stub is labeled by the degree of the neighbor vertex incident to the edge whose cut resulted in the stub, the sum of these labels is the sum of the neighbor degrees. In the next section, we give necessary and sufficient conditions when a sequence of labeled stub-stars has graph realizations with certain properties. Our main interest is in the cases when the graph is a simple graph or a cycle-free graph (tree or forest) or a caterpillar. However, for the sake of completeness, we also discuss the cases when the graph is a multigraph or a loopless multigraph. After this, we show that for each of the five cases (multigraph, loopless multigraph, simple graph, cycle-free graph, caterpillar) a polynomial time solvable integer programming feasibility problem exists to find a sequence of stub-stars that is consistent with a given degree and neighbor degree sequence and also satisfies the necessary conditions to have a graph realization with the given properties.

2. Preliminaries

First, we give some graph-theoretical definitions.

Definition 1. A graph $G = (V, E)$ is a *multigraph* if E is a multiset of $\binom{V}{2} \cup \binom{V}{1}$. G is a *loopless multigraph* if E is a multiset of $\binom{V}{2}$. G is a *simple graph* if E is a subset of $\binom{V}{2}$.

A *caterpillar* is a tree in which the non-leaf vertices form a path.

A *component* of a (multi)graph is a connected subgraph that is not part of any larger connected subgraph.

Definition 2. A degree sequence $(d_1, d_2, \ldots, d_n)$ is *graphical* if there exists a vertex labeled graph $G = (V, E)$ such that for each $v_i \in V$, $d(v_i) = d_i$. Such a graph G is called a *realization* of the degree sequence $(d_1, d_2, \ldots, d_n)$.

A pair of degree sequences (or *bipartite degree sequence*) $(d_{1,1}, d_{1,2}, \ldots, d_{1,n})$, $(d_{2,1}, d_{2,2}, \ldots, d_{2,m})$ is *graphical* if there exists a simple bipartite graph $G = (U, V, E)$, such that for each $u_i \in U$, $d(u_i) = d_{1,i}$, and for each $v_j \in V$, $d(v_j) = d_{2,j}$. Such a bipartite graph G is also called a realization.

A degree sequence or bipartite degree sequence is *forest realizable* if it has a forest realization, that is, a realization which is a forest.

Definition 3. Let $D = (d_1, d_2, \ldots, d_n)$ and $F = (f_1, f_2, \ldots, f_n)$ be a pair of degree and neighbor degree sequences. We say that the graph $G = (V, E)$ is a *realization* of (D, F) if for all k, $d(v_k) = d_k$ and $\sum_{w \in \mathcal{N}(v_k)} d(w) = f_k$, where $\mathcal{N}(v)$ denotes the set of neighbors of v.

We now introduce the main technical concept used in this paper: the labeled stub-stars.

Definition 4. A *labeled stub-star* S_k of *degree* d is a vertex with d stubs (half edges). Each stub s_i is labeled with a positive integer $d_{k,i}$. The *degree* of a labeled stub-star is the number of its stubs. The *neighbor degree sum* of a labeled stub-star is the sum of its labels. Let $\mathcal{S} = (S_1, S_2, \ldots, S_n)$ be a sequence of labeled stub-stars. The *maximum degree* of a sequence of labeled stub-stars is the maximum degree of its labeled stub-stars. We denote it by Δ.

The labels of a labeled stub-star form a partition of the sum of the labels. We will identify labeled stub-stars by these partitions. We will write the numbers in a partition in decreasing order, and put them into square brackets.

A vertex labeled graph $G = (V, E)$ is a *realization* of $\mathcal{S}$ if for each $v_k \in V$, the neighbors of v_k have degrees $d_{k,1}, d_{k,2}, \ldots, d_{k,d(v_k)}$, where $d_{k,1}, d_{k,2}, \ldots, d_{k,d(v_k)}$ are the labels of the stub-star S_k. We say that $\mathcal{S}$ is multigraph, loopless multigraph, simple graph, cycle-free graph, caterpillar realizable, respectively, if it has a realization which is a multigraph, loopless multigraph, simple graph, cycle-free graph, caterpillar, respectively.

Definition 5. The *chromatic degree* $d_{(i,j)}$ $(i \le j)$ of a labeled stub-star S is 0 if the degree of S is neither i nor j. If its degree is i, then $d_{(i,j)}(S)$ is the number of times S contains label j. Finally, if the degree of S is j, then $d_{(i,j)}(S)$ is the number of times S contains label i.

For a sequence of labeled stub-stars $\mathcal{S} = (S_1, S_2, \ldots, S_n)$, we define a degree sequence

$$D_{(i,i)}(\mathcal{S}) := (d_{(i,i)}(S_1), d_{(i,i)}(S_2), \ldots, d_{(i,i)}(S_n)).$$

We also define the bipartite degree sequence $D_{(i,j)}(\mathcal{S})$ for all $i < j$. First we split $\mathcal{S}$ into two sequences. Let $(A_1, A_2, \ldots, A_r)$ be the subsequence of $\mathcal{S}$ of stub stars with degree i and let $(B_1, B_2, \ldots, B_s)$ be the subsequence of stub stars with degree not equal to i. Then

$$D_{(i,j)}(\mathcal{S}) := (d_{(i,j)}(A_1), d_{(i,j)}(A_2), \ldots, d_{(i,j)}(A_r)), (d_{(i,j)}(B_1), d_{(i,j)}(B_2), \ldots, d_{(i,j)}(B_s)).$$

For a graph G, we define the *monochromatic sub-graph* of G with color (i, j) to be the subgraph containing the edges that connect vertices with degree i and j.

As an example, a realization of the sequence of labeled stub-stars

$$\mathcal{S} = ([3, 2], [3, 2], [3], [2, 2, 1], [3], [3], [3, 1, 1], [3, 1, 1], [3], [3])$$

is given in Fig. 1. The chromatic degree sequences can be easily obtained by considering the monochromatic subgraphs containing only the edges with the given color. For example, its $D_{(2,2)}$ degree sequence is

$$D_{(2,2)}(\mathcal{S}) = (1, 1, 0, 0, 0, 0, 0, 0, 0, 0),$$

and its $D_{(1,3)}$ bipartite degree sequence is

$$D_{(1,3)}(\mathcal{S}) = (1, 1, 1, 1, 1), (0, 0, 1, 2, 2).$$

The labels of a labeled stub-star form a partition. Below we give the necessary definitions and notations for partitions of integer numbers used in this paper.

Definition 6. We will denote by $\lambda \dashv n$ that $\lambda = [\lambda_1, \lambda_2, \ldots, \lambda_k]$ is a partition of a positive integer n. That is, each λ_i is a positive integer, and it holds that

$$\sum_{i=1}^{k} \lambda_i = n.$$

The *height* of $\lambda = [\lambda_1, \lambda_2, \ldots, \lambda_k]$ is k, and is denoted by $h(\lambda)$. The *size* of λ is n, and is denoted by $|\lambda|$. Finally, $N(\lambda, j)$ denotes how many times j appears in λ, that is

$$N(\lambda, j) := |\{i \in [h(\lambda)] | \lambda_i = j\}|.$$

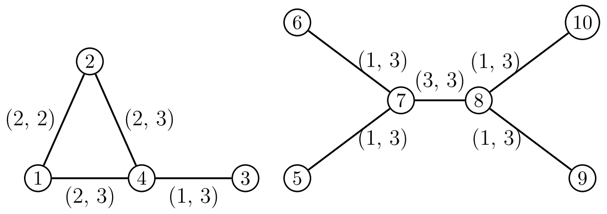


Fig. 1. A realization of the sequence $\mathcal{S} = ([3, 2], [3, 2], [3], [2, 2, 1], [3], [3], [3, 1, 1], [3, 1, 1], [3], [3])$ of labeled stub-stars. The vertices are labeled, and also, the colors of the edges are indicated.

If λ comes from the labels of a stub-star, then $h(\lambda)$ corresponds to the degree of the stub-star while $|\lambda|$ corresponds to the neighbor degree sum, that is, the sum of the labels of the stubs.

We are going to solve the graph realization problems defined above via integer programming feasibility problems that we introduce now.

Definition 7. The *integer programming feasibility problem* $(\mathcal{X}, \mathcal{L})$ consists of a set $\mathcal{X}$ of variables, and a system $\mathcal{L}$ of linear inequalities whose variables are from $\mathcal{X}$ and asks if there is an integer assignment to $\mathcal{X}$ that satisfies $\mathcal{L}$.

Some of the inequalities we are going to use contain the function $\min\{x, y\}$ or $\max\{x, y\}$. These can be expressed as linear inequality systems if an upper bound for $|x - y|$ is available. The following lemma is well known in operational research, but for the sake of completeness, we give a proof here.

Lemma 8. *Let x, y and M be integer numbers satisfying*

$$M \geq |x - y|,$$

and let z and b be variables. Then the following inequality system

$$z \leq x \tag{1}$$
$$z \leq y \tag{2}$$
$$z \geq x - Mb \tag{3}$$
$$z \geq y - M(1 - b) \tag{4}$$
$$0 \leq b \tag{5}$$
$$b \leq 1 \tag{6}$$

has at most two integer solutions, and in every case $z = \min\{x, y\}$.

Similarly, the following inequality system

$$z \geq x \tag{7}$$
$$z \geq y \tag{8}$$
$$z \leq x + Mb \tag{9}$$
$$z \leq y + M(1 - b) \tag{10}$$
$$0 \leq b \tag{11}$$
$$b \leq 1 \tag{12}$$

has at most two integer solutions, and in every case $z = \max\{x, y\}$.

Proof. We prove the $\min\{x, y\}$ case. Observe that b is either 0 or 1. If $x < y$, then there is no solution with $b = 1$ since the inequalities in Eqs. (1) and (4) would contradict each other. On the other hand, $b = 0$ and $z = x$ is a solution, and the only solution since $x \leq z \leq x$ must hold.

Similarly, when $y < x$, the only solution is $z = y$, $b = 1$. Finally, if $x = y$, then $z = x (= y)$ and $b \in \{0, 1\}$.

The proof for $\max\{x, y\}$ goes analogously. □

3. Realizations of a set of labeled stub-stars

In this section, we state and prove theorems on realizations of labeled stub-stars. We need the well known Erdős–Gallai and Gale–Ryser inequalities.

Theorem 9 (*Erdős–Gallai, [6]*). *Let $D = (d_1 \geq d_2 \geq \cdots \geq d_n)$ be a degree sequence. Then D is simple graph realizable if and only if the sum of the degrees is even, and for each $k = 1, 2, \ldots, n$, the inequality*

$$\sum_{g=1}^{k} d_g \leq k(k-1) + \sum_{h=k+1}^{n} \min\{k, d_h\} \tag{13}$$

holds.

Theorem 10 (*Gale–Ryser, [8,16]*). *Let $D = (d_{1,1} \geq d_{1,2} \geq \cdots \geq d_{1,n}), (d_{2,1}, d_{2,2}, \ldots, d_{2,m})$ be a bipartite degree sequence. Then D is bipartite graph realizable if and only if*

$$\sum_{g=1}^{n} d_{1,g} = \sum_{h=1}^{m} d_{2,h} \tag{14}$$

and for each $k = 1, 2, \ldots, n$, the inequality

$$\sum_{g=1}^{k} d_{1,g} \leq \sum_{h=1}^{m} \min\{k, d_{2,h}\} \tag{15}$$

holds.

When the degree sequences have a maximum degree Δ, then only the first Δ Erdős–Gallai and the first $\Delta - 1$ Gale–Ryser inequalities have to be checked as the following well known lemma states.

Lemma 11. *Let $D = (d_1 \geq d_2 \geq \cdots \geq d_n)$ be a degree sequence. Then for each $k > d_1$, the inequality in Eq. (13) holds. Similarly, let $D = (d_{1,1} \geq d_{1,2} \geq \cdots \geq d_{1,n}), (d_{2,1}, d_{2,2}, \ldots, d_{2,m})$ be a bipartite degree sequence satisfying equation (14). Then for all $k \geq \max_j\{d_{2,j}\}$, the inequality in Eq. (15) holds.*

For sake of completeness, we mention the following well-known theorems.

Theorem 12. *A degree sequence $D = (d_1, d_2, \ldots, d_n)$ is tree-realizable if and only if all degrees are positive and $\sum_{i=1}^{n} d_i = 2n - 2$. D is forest-realizable if and only if there are exactly m non-zero degrees, and their sum is even and at most $2m - 2$. A degree sequence is caterpillar realizable if and only if it is tree realizable.*

In the following theorem and later in the paper, we use conditional sums. Conditions are introduced after a '|'. For example, '$\lambda | h(\lambda) = i$' means that the summation is only for those partitions λ whose height is i.

Theorem 13. *Let $\mathcal{S} = (S_1, S_2, \ldots, S_n)$ be a sequence of labeled stub-stars with maximum degree Δ. Let x_λ denote the number of labeled stub-stars in $\mathcal{S}$ whose labels form the partition λ. Then $\mathcal{S}$ is*

(a) multigraph
(b) loopless multigraph
(c) simple graph

realizable if and only if

(a) for each $i = 1, 2, \ldots, \Delta$,

$$\sum_{\lambda | h(\lambda) = i} x_\lambda N(\lambda, i) \equiv 0 \pmod 2 \tag{16}$$

and for each $1 \leq i < j \leq \Delta$

$$\sum_{\lambda | h(\lambda) = i} x_\lambda N(\lambda, j) = \sum_{\lambda' | h(\lambda') = j} x_{\lambda'} N(\lambda', i). \tag{17}$$

(b) the conditions in case (a) holds and for each $i = 1, 2, \ldots, \Delta$,

$$2 \max_{\lambda | h(\lambda) = i} \{\min\{1, x_\lambda\} N(\lambda, i)\} \leq \sum_{\lambda' | h(\lambda') = i} x_{\lambda'} N(\lambda', i). \tag{18}$$

(c) the conditions in case (a) holds, for each $i = 1, 2, \ldots, \Delta$, the first Δ Erdős–Gallai inequalities hold for $D_{(i,i)}(\mathcal{S})$, and for each $i < j$, $i, j = 1, 2, \ldots, \Delta$, the first $\Delta - 1$ Gale–Ryser inequalities hold for the bipartite degree sequence $D_{(i,j)}(\mathcal{S})$.

Proof.

(a) It is well known that a degree sequence $(d_1, d_2, \dots, d_n)$ has a multigraph realization if and only if $\sum_{i=1}^{n} d_i$ is even. Similarly, the bipartite degree sequence $(d_{1,1}, d_{1,2}, \dots, d_{1,n}), (d_{2,1}, d_{2,2}, \dots, d_{2,m})$ has a bipartite multigraph realization if and only if $\sum_{i=1}^{n} d_{1,i} = \sum_{j=1}^{m} d_{2,j}$. Observe that Eq. (16) says that the sum of the degrees in $D_{(i,i)}(\mathcal{S})$ is even, furthermore, Eq. (17) says that the sum of the degrees in the two sequences are the same in the bipartite degree sequence $D_{(i,j)}(\mathcal{S})$.
Let G be a multigraph realization of $\mathcal{S}$. Then for each $i = 1, 2, \dots, \Delta$, the monochromatic subgraph with color (i, i) is a multigraph, and for each $i < j$, $i, j = 1, 2, \dots, \Delta$, the monochromatic subgraph with color (i, j) is a bipartite multigraph. That is, Eqs. (16) and (17) hold.
On the other hand, if Eqs. (16) and (17) hold, then there are multigraph realizations for each $D_{(i,i)}(\mathcal{S})$ and there are bipartite multigraph realizations for each $D_{(i,j)}(\mathcal{S})$. The union of them is a multigraph realization of $\mathcal{S}$.

(b) It is also well known that a degree sequence $(d_1, d_2, \dots, d_n)$ has a loopless multigraph realization if and only if the sum of the degrees is even and

$$\max_i\{d_i\} \leq \sum_{i=1}^{n} d_i - \max_i\{d_i\}.$$

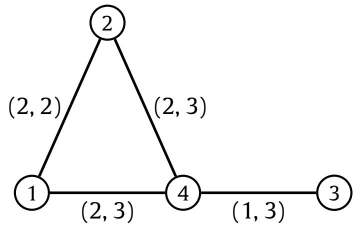
Observe that Eq. (18) describes this condition for the degree sequence $D_{(i,i)}(\mathcal{S})$.
Let G be a loopless multigraph realization of $\mathcal{S}$. Then for each $i = 1, 2, \dots, \Delta$, the monochromatic subgraph with color (i, i) is a loopless multigraph, and for each $i < j$, $i, j = 1, 2, \dots, \Delta$, the monochromatic subgraph with color (i, j) is a bipartite (loopless) multigraph. (Observe that bipartite graphs cannot contain a loop.) That is, Eqs. (16), (17) and (18) hold.
On the other hand, if Eqs. (16), (17) and (18) hold, then there are loopless multigraph realizations for each $D_{(i,i)}(\mathcal{S})$ and there are bipartite multigraph realizations for each $D_{(i,j)}(\mathcal{S})$. The union of them is a loopless multigraph realization of $\mathcal{S}$.

(c) Let G be a simple graph realization of $\mathcal{S}$. Then for each color (i, i), $i = 1, 2, \dots, \Delta$, the monochromatic subgraph is a simple realization of the degree sequence $D_{(i,i)}(\mathcal{S})$, and for each color (i, j), $i < j$, $i, j = 1, 2, \dots, \Delta$, the monochromatic subgraph is a simple bipartite realization of the bipartite degree sequence $D_{(i,j)}(\mathcal{S})$. Therefore the conditions in case (a) as well as the first Δ Erdős–Gallai and the first $\Delta - 1$ Gale–Ryser inequalities hold.
On the other hand, if the conditions in case (a) as well as the first Δ Erdős–Gallai and the first $\Delta - 1$ Gale–Ryser inequalities hold, then there are monochromatic simple graph realizations of each color (i, i), and there are monochromatic simple bipartite graph realizations for each color (i, j), $i < j$, $i, j = 1, 2, \dots, \Delta$. Observe that the union of them is a simple graph realization of $\mathcal{S}$. Indeed, no parallel edges are possible in the union of these monochromatic graphs. Any parallel edge in the union would be a parallel edge in one of the monochromatic graphs, but they are all simple. □

We now consider when a sequence of labeled stub-stars has a forest realization. Given the previous theorem, we might hope that a sufficient condition is if for each $i \leq j$, $i, j = 1, 2, \dots, \Delta$, $D_{(i,j)}$ is a forest realizable degree sequence. However, this is not the case, as the following example demonstrates.

The example sequence of labeled stub-stars $\mathcal{S}$ consists of four labeled stub-stars, S_1, S_2, S_3, S_4, respectively, with labels forming partitions [3, 2], [3, 2], [3], [2, 2, 1], respectively.

In this case, each $D_{(i,i)}(\mathcal{S})$ and $D_{(i,j)}(\mathcal{S})$ is forest realizable, but $\mathcal{S}$ has only one realization, shown below.



The issue is that unlike loops and parallel edges, cycles can occur among multiple colors, so checking each individual degree sequence is not sufficient. This motivates the next definition.

Definition 14. Let $\mathcal{S} = (S_1, S_2, \dots, S_n)$ be a sequence of labeled stub-stars with maximum degree Δ. For any subset $I \subseteq \{(i, j) | 1 \leq i \leq j \leq \Delta\}$, we define the *multichromatic degree* of S_k to be $d_I(S_k) := \sum_{(i,j)\in I} d_{(i,j)}(S_k)$ and the *multichromatic degree sequence* to be $D_I(\mathcal{S}) := (d_I(S_1), d_I(S_2), \dots, d_I(S_n))$.

For example, let $I = \{(2, 2), (2, 3)\}$ for the previous example $\mathcal{S}$. Then $d_I(S_1) = 2, d_I(S_2) = 2, d_I(S_3) = 0$, and $d_I(S_4) = 2$. We also would like to warn the readers that multichromatic degree sequences differ from monochromatic degree sequences even if the set I contains a single color. For example, for the set of labeled stub-stars $\mathcal{S}$ whose realization was shown in Fig. 1, the bipartite degree sequence for the color (1, 3) was $D_{(1,3)}(\mathcal{S}) = (1, 1, 1, 1, 1), (0, 0, 1, 2, 2)$. However, the multichromatic degree sequence for $I = \{(1, 3)\}$ is $D_{\{(1,3)\}}(\mathcal{S}) = (0, 0, 1, 1, 1, 1, 2, 2, 1, 1)$.

Theorem 15. *Let $\mathcal{S} = (S_1, S_2, \ldots, S_n)$ be a sequence of labeled stub-stars with maximum degree Δ. $\mathcal{S}$ is forest realizable if and only if it is multigraph realizable and for any subset $I \subseteq \{(i,j)|1 \leq i \leq j \leq \Delta\}$, the multichromatic degree sequence $(d_I(S_1), d_I(S_2), \ldots, d_I(S_n))$ is forest realizable.*

Before we give the proof, we note that multigraph realizability is not a consequence of the condition that for all subsets of colors, the multichromatic degree sequence is forest realizable. The example is the following sequence of labeled stub-stars:

$$\mathcal{S} = ([3, 1, 1, 1], [4, 1, 1], [4, 1, 1], [4, 1, 1], [4], [4], [4], [3], [3], [3], [3], [3], [3]).$$

This sequence of labeled stub-stars is clearly not multigraph-realizable, as there is only one degree 4 stub-star, however, there are 6 stubs with label 4. However, for all subsets of colors, the multichromatic degree sequence is forest realizable. The key is that the monochromatic degree sequence for color $(3, 4)$ is the bipartite degree sequence

$$(1), (1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0)$$

which is not bipartite graph realizable. On the other hand, the multichromatic degree sequence for $I = \{(3, 4)\}$ is $(1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0)$ which is forest realizable.

Proof of Theorem 15. We first show the conditions are necessary. Let G be a forest realization of $\mathcal{S}$. Then G is also a multigraph realization, so $\mathcal{S}$ is multigraph realizable. For any subset of colors I, we define the color-induced subgraph of G, denoted $G[I]$, in the following way. For each edge, we label the edge with color (i, j) if the edge is created by joining two stubs with labels i and j. Then $G[I]$ is the subgraph consisting of all vertices of G along with all edges labeled with some element of I. Since G is a forest, then $G[I]$ is also a forest. The degree sequence of $G[I]$ is $(d_I(S_1), d_I(S_2), \ldots, d_I(S_n))$, so this sequence must be forest realizable.

To show the conditions are sufficient, we consider a multigraph realization G which is minimal among all realizations of $\mathcal{S}$ in the number of components. By assumption multigraph realizations exist and there are a finite number of them, so there exists a realization with minimal number of components.

Assume to the contrary that G has a cycle or a loop. Since we start with a multigraph realization, a cycle might have a length of two, that is, two vertices connected by two parallel edges. In the case that G contains a cycle or a loop, we will produce another realization of S, which we will call G', with one fewer components. First we construct a subgraph H_k of G, then we will find a series of swap operations that lead to G' with one fewer components.

Let $C := C_0$ be a cycle or a loop, and let I_0 be the set of colors of its edges. Let H_0 be $G[I_0]$. H_0 cannot have only one component as it would contradict that D_{I_0} is forest realizable. Indeed, a connected graph with a cycle has a degree sequence that is not forest realizable. However, it can happen that H_0 is contained in one component of G. Therefore, we define two sequences H_i, C_i of subgraphs of G and a sequence of subsets of colors, I_i starting with $i = 0$. While H_i is contained in one component of G, we construct an H_{i+1}, a C_{i+1} and I_{i+1} in the following way:

1. For all pairs of edges (e_1, e_2) such that e_1 and e_2 have the same color in I_i, e_1 is in C_i and e_2 is in another component of H_i, find a path from e_1 to e_2 in G. Such path exists as H_i is contained in one component of G. For each edge f in that path whose color is not in I_i, add its color to I_{i+1}, and label f by (e_1, e_2). An edge might be in several paths for several (e_1, e_2) pairs, in that case take any of these labels.
2. Add all the colors in I_i to I_{i+1}. Then H_{i+1} is defined as $G[I_{i+1}]$. Note that H_{i+1} is disconnected because it has a cycle yet $D_{I_{i+1}}$ is forest realizable. Define C_{i+1} as the component of H_{i+1} that contains C_i.

In a finite number of steps, this procedure will create an H_k which is contained in more than one component of G. Indeed, note that H_i created by the procedure is always disconnected (this is since the degree sequence D_{I_i} is forest realizable and H_i contains a cycle). At the same time, the inclusion of H_i in H_{i+1} is always proper (indeed, as long as H_i is contained in one component of G, we have $I_i \subset I_{i+1}$). Recall that G is disconnected (this is since its degree sequence is forest realizable and G contains a cycle). Consequently, the procedure will finally create H_k which is contained in more than one component of G. Since the set of colors is finite, the procedure will terminate in a finite number of steps.

Now we are going to construct a G' that has one fewer component than G. Let $f_{1,k}$ and $f_{2,k}$ be two edges with the same color in $I_k \setminus I_{k-1}$ such that $f_{1,k}$ is in C_k and $f_{2,k}$ is not in the same component of G as $f_{1,k}$. Assume that $f_{1,k} = (v_1, v_2)$, $f_{2,k} = (v_3, v_4)$, and v_1 and v_4 have the same degree. Furthermore, let us call the two components (in G) containing $f_{1,k}$ and $f_{2,k}$ by G_1 and G_2.

Now we remove $f_{1,k}$ and $f_{2,k}$ from G and add edges $f_1' = (v_1, v_3)$ and $f_2' = (v_2, v_4)$. This *swap operation* creates another simple graph realization of $\mathcal{S}$ from G. If $f_{1,k}$ or $f_{2,k}$ is in a cycle in G, then the swap operation merges G_1 and G_2 of G into one component, thus, we arrive at a realization G' of S with one fewer component. Otherwise the swap operation does not change the number of components. Indeed, in that case, both removing $f_{1,k}$ and $f_{2,k}$ splits their components, G_1 and G_2 in G into two components, $G_{1,a}$, $G_{1,b}$ and $G_{2,a}$, $G_{2,b}$; however, the two new edges connects $G_{1,a}$ to $G_{2,a}$ and $G_{1,b}$ to $G_{2,b}$.

On the other hand, consider the label $(f_{1,k-1}, f_{2,k-1})$ of $f_{1,k}$ if the swap operation does not decrease the number of components. We claim the following: if the swap operation does not decrease the number of components in G then $f_{1,k-1}$ and $f_{2,k-1}$ are in different components of G after the swap operation. Indeed, we know that there was a path between $f_{1,k-1}$ and $f_{2,k-1}$ before the swap operation that contained $f_{1,k}$. If there is a path between $f_{1,k-1}$ and $f_{2,k-1}$ after the swap operation, then either this path is in G_1, and in that case $f_{1,k}$ was in a cycle, a contradiction, or this path contains edges in G_2, but in that case $f_{2,k}$ was in a cycle, also a contradiction (or both).

Now we can also perform a swap operation using edges $f_{1,k-1}$ and $f_{2,k-1}$. This operation either decreases the number of components, and then we arrive at G' with 1 fewer component, or separates $f_{1,k-2}$ and $f_{2,k-2}$, the two edges in the label of $f_{1,k-1}$. We can iterate this process since all the swap operations separating $f_{1,i}$ and $f_{2,i}$ use edges with color in $I_k \setminus I_i$, and thus do not change any edge in the paths between $f_{1,j}$ and $f_{2,j}$ for any $j < i$.

Eventually, one of the swap operations will merge two components in G since $f_{1,0}$ is in the cycle or loop C. We remark here if C is a loop, then the swap operation removes the loop (v, v) and edge (u, w) and adds the edges (v, u) and (v, w).

We arrived at a contradiction that G had the minimum number of components. Therefore, G does not have a cycle, thus it is a forest realization. □

Observe that the proof also provides an algorithmic construction. Take any multigraph realization and if it contains a loop or cycle, then the construction in the proof provides a series of swap operations decreasing the number of components in the realization. We remark that a series of swap operations given by the proof leads to a realization (called G') with one fewer component. Eventually, after a finite number of swap operations (possibly via several G's), a forest realization is constructed.

Finally, we give the necessary and sufficient conditions when a sequence of labeled stub-stars are caterpillar realizable.

Theorem 16. *Let $\mathcal{S} = (S_1, S_2, \dots, S_n)$ be a sequence of labeled stub-stars, such that none of the stub-stars has* 0 *degree. Then $\mathcal{S}$ has a caterpillar realization if it is forest realizable, the sum of the degrees is $2n - 2$, and there is no labeled stub-star in $\mathcal{S}$ with more than two labels larger than* 1.

Proof. First we show that the conditions are necessary. Let G be a caterpillar realization of $\mathcal{S}$. Then $\mathcal{S}$ is clearly forest-realizable, and the sum of the degrees of the labeled stub-stars is $2n - 2$. Also, in a caterpillar realization, any vertex has at most two neighbors with degree larger than 1.

Now we show that the conditions are sufficient. If $\mathcal{S}$ is forest realizable, take any forest realization of it, G. We show that G is a caterpillar. First, G is a tree since the sum of the degrees is $2n - 2$. Furthermore, it is a caterpillar since there is no vertex with more than two neighbors whose degrees are larger than 1. □

4. Degree and neighbor degree constraint realizations

In the previous section, we gave necessary and sufficient conditions when a sequence of labeled stub-stars have realizations with given properties. In this section, we show how those results fit into an integer programming feasibility problem. Recall that an integer programming feasibility problem asks if a bunch of linear inequalities have an integer solution. Although the integer programming feasibility problem in general is NP-complete, here we will have constant number of variables and equations. That is, both the number of variables and the number of inequalities depend only on Δ and not on n. Therefore, for each mentioned problem, the same integer programming feasibility problem instance is solved for any problem instance, just with different constants. The constants grow linearly with the number of vertices, thus the running time of the integer programming feasibility grows only with a poly-log function of the number of vertices.

First, we give the equation system that defines the possible sequences of labeled stub-stars whose realizations are realizations of a given degree and neighbor degree sequence.

Lemma 17. *Let $D = (d_1, d_2, \dots, d_n)$ and $F = (f_1, f_2, \dots, f_n)$ be a pair of degree and neighbor degree sequences. Then a graph $G = (V, E)$ is a realization of (D, F) if and only if G is a realization of $\mathcal{S} = (S_1, S_2, \dots, S_n)$ satisfying the following inequality system:*

$$\sum_{\lambda \mid \lambda \dashv f \ \wedge \ h(\lambda) = d} x_\lambda = y_{d,f} \tag{19}$$

$$x_\lambda \geq 0 \tag{20}$$

where x_λ is the number of labeled stub-stars in $\mathcal{S}$ with labels λ (recall that the labels of a labeled stub-star form a partition of a positive integer) and $y_{d,f}$ is the number of cases when $d_i = d$ and $f_i = f$ (d_i and f_i in the sequences D and F).

Proof. Assume that $G = (V, E)$ is a realization of (D, F). Then cutting each edge in E and labeling the stubs of the so-emerging stub-stars by the degree of the neighbor vertices yields a labeled stub-star S for each vertex $v \in V$. The degree of S is $d(v)$ and its labels form a partition of $f := \sum_{w \in \mathcal{N}(v)} d(w)$. Therefore any realization of (D, F) is also a realization of a sequence of labeled stub-stars that satisfies the equation system summarized in Eq. (19).

Similarly, if $G = (V, E)$ is a realization of a sequence of labeled stub-stars satisfying the equation system presented in Eq. (19), then the degree sequence of G is D, and the neighbor degree sum sequence is F. □

We can combine the equation system given in Eqs. (19) and (20) with appropriate linear inequality systems describing the conditions (a)–(c) in Theorem 13 and the conditions in Theorems 15 and 16.

First, we start with multigraph realizations.

Theorem 18. *Let $D = (d_1, d_2, \ldots, d_n)$ and $F = (f_1, f_2, \ldots, f_n)$ be a pair of degree and neighbor degree sequences. Let Δ denote the largest degree in D. Then (D, F) is multigraph realizable if and only if the integer programming feasibility problem $\{\mathcal{X}, \mathcal{L}\}$ has a solution, where $\mathcal{X}$ consists of all x_λ variables for all partitions $\lambda = [\lambda_1, \lambda_2, \ldots, \lambda_{h(\lambda)}]$ such that $h(\lambda) \leq \Delta$ and for all i, $\lambda_i \leq \Delta$ and auxiliary variables $p_1, p_2, \ldots, p_\Delta$, and $\mathcal{L}$ consists of*

- *the inequality system given in Eqs.* (19) and (20)*,*
- *for all $0 < i < j \leq \Delta$, the equation*

$$\sum_{\lambda | h(\lambda)=i} x_\lambda N(\lambda, j) = \sum_{\lambda' | h(\lambda')=j} x_{\lambda'} N(\lambda', i), \tag{21}$$

- *and for all i, the equation*

$$\sum_{\lambda | h(\lambda)=i} x_\lambda N(\lambda, i) = 2p_i. \tag{22}$$

Proof. Assume that (D, F) has a multigraph realization G. Decompose G into labeled stub-stars $\mathcal{S}$, then G is a multigraph realization of $\mathcal{S}$.

Let x_λ denote the number of labeled stub-stars in $\mathcal{S}$ whose labels are λ. Furthermore, let

$$p_i := \frac{1}{2} \sum_{\lambda | h(\lambda)=i} x_\lambda N(\lambda, i).$$

We claim that these assignments form a solution to $(\mathcal{X}, \mathcal{L})$. Indeed, the inequality system in Eqs. (19) and (20) holds due to Lemma 17. Eqs. (21) and (22) hold due to Theorem 13. Indeed, Eq. (21) shows that for each $i < j$, the degree sequence $D_{(i,j)}(\mathcal{S})$ has a bipartite graph realization where all edges go between centers of stub-stars with degree i and j. Furthermore, Eq. (22) shows that for each i, the sum of the degrees in $D_{(i,i)}(\mathcal{S})$ is even. Therefore, if (D, F) has a multigraph realization, the integer programming feasibility problem $(\mathcal{X}, \mathcal{L})$ has a solution.

Now assume that $(\mathcal{X}, \mathcal{L})$ has a solution. Let $\mathcal{S}$ denote the sequence of labeled stub-stars in which the number of labeled stub-stars with labels λ is x_λ. Since Eqs. (21) and (22) hold, $\mathcal{S}$ has a multigraph realization G according to Theorem 13. This realization G is a realization of (D, F) by Lemma 17 since the inequality system given in Eqs. (19) and (20) hold. □

For loopless multigraphs, a similar theorem can be proved.

Theorem 19. *Let $D = (d_1, d_2, \ldots, d_n)$ and $F = (f_1, f_2, \ldots, f_n)$ be a pair of degree and neighbor degree sequences. Let Δ denote the largest degree in D. Then (D, F) is loopless multigraph realizable if and only if the integer programming feasibility problem $\{\mathcal{X}, \mathcal{L}\}$ has a solution, where $\mathcal{X}$ consists of all x_λ variables for all partitions $\lambda = [\lambda_1, \lambda_2, \ldots, \lambda_{h(\lambda)}]$ such that $h(\lambda) \leq \Delta$ and for all i, $\lambda_i \leq \Delta$ and auxiliary variables $p_1, p_2, \ldots, p_\Delta, z_1, z_2, \ldots, z_r, b_1, b_2, \ldots, b_r$, where $r = \sum_{i=1}^{\Delta}(2i - 1)$ and $\mathcal{L}$ consists of*

- *the inequality system given in Eqs.* (19) and (20)*,*
- *for all $0 < i < j \leq \Delta$, the equation*

$$\sum_{\lambda | h(\lambda)=i} x_\lambda N(\lambda, j) = \sum_{\lambda' | h(\lambda')=j} x_{\lambda'} N(\lambda', i), \tag{23}$$

- *for all i, the equation*

$$\sum_{\lambda | h(\lambda)=i} x_\lambda N(\lambda, i) = 2p_i, \tag{24}$$

- *and for all i, the integer linear programming feasibility problem defining*

$$z_{s(i)} = \max_{k \in \{1,2,\ldots,i\}} \left\{ \min \left\{ 1, \sum_{\lambda | h(\lambda)=i \wedge N(\lambda,i)=k} x_\lambda \right\} k \right\}, \tag{25}$$

  where $s(i)$ is the index of the auxiliary variable storing the maximum degree in $D_{(i,i)}(\mathcal{S})$ as well as the inequality

$$2z_{s(i)} \leq \sum_{\lambda | h(\lambda)=i} x_\lambda N(\lambda, i). \tag{26}$$

Proof. Similar to the proof of Theorem 18. Observe that the linear programming described in Eqs. (25) and (26) is equivalent with the inequality in Eq. (18). Indeed, the maximum degree is the largest k for which $\sum_{\lambda | h(\lambda)=i \wedge N(\lambda,i)=k} x_\lambda$ is not 0. There are i minimums in Eq. (25). We need i auxiliary z and b variables for them. Then we have to select the maximum of the i variables. It can be done by $i-1$ pairwise maximization. Thus, for one i, there are $2i-1$ maximization/minimizations, needing $2i - 1$ z's and b's. This explains that the number of auxiliary variables, r is $\sum_{i=1}^{\Delta}(2i - 1)$. □

In case of simple graph realizations, the first few Erdős–Gallai and Gale–Ryser inequalities should be checked. The challenge is that the degrees in $D_{(i,j)}(\mathcal{S})$ and $D_{(i,i)}(\mathcal{S})$ are not ordered so we do not know what the first k highest degrees are. Fortunately, it is possible to give a dynamic programming algorithm that computes the sum of the first k highest degrees. Below we introduce a dynamic programming algorithm that computes $t_{i,k}$ for $k = 1, 2, \ldots, \Delta$, which are the auxiliary variables that count the sum of the k largest degrees needed for Erdős–Gallai inequalities and the values $t_{l,k}$ for $l < i$ which are needed for recursive derivation of the $t_{i,k}$'s. The number of arithmetic operations in the dynamic programming algorithm depends only on Δ.

Lemma 20. *Let $\mathcal{S}$ be a sequence of labeled stub-stars in which the number of labeled stub-stars with labels λ is x_λ. Fix an i and j, and let*

$$s_l := \sum_{\lambda | h(\lambda)=i \wedge N(\lambda,j)=l} x_\lambda .$$

Let $t_{l,k}$ denote the largest possible sum of at most k entries in $D_{(i,j)}(\mathcal{S})$ corresponding to labeled stub-stars with degree i such that each term in the sum is at most l. Then the equations

$$t_{l,0} = 0 \tag{27}$$

and

$$t_{l,k} = \max_{0 \le k' \le k} \left\{ t_{l-1,k-k'} + l \times \min\{k', s_l\} \right\} \quad \forall\, l > 1 \tag{28}$$

hold.

Proof. Eq. (27) is trivial. We prove Eq. (28) by proving two inequalities. To prove that the left hand side is smaller than or equal the right hand side, consider the sum that maximizes $t_{l,k}$. Assume that it contains $k - \tilde{k}$ terms that are smaller than l. Then

$$t_{l,k} \le t_{l-1,k-\tilde{k}} + l \times \min\{\tilde{k}, s_l\}$$

since the sum of these $k - \tilde{k}$ terms cannot be larger than $t_{l-1,k-\tilde{k}}$. Thus,

$$t_{l,k} \le t_{l-1,k-\tilde{k}} + l \times \min\{\tilde{k}, s_l\} \le \max_{0 \le k' \le k} \left\{ t_{l-1,k-k'} + l \times \min\{k', s_l\} \right\} .$$

To prove that the left hand side is greater than or equal the right hand side, consider the $\tilde{k}$ that maximizes the right hand side, and consider the sum that maximizes $t_{l-1,k-\tilde{k}}$. Add $\min\{\tilde{k}, s_l\}$ times l to this sum, then we get a sum that contains at most k terms, the largest term is at most l, and its value is $t_{l-1,k-k'} + l \times \min\{k', s_l\}$. It cannot be larger than $t_{l,k}$, thus we get that

$$t_{l,k} \ge t_{l-1,k-\tilde{k}} + l \times \min\{\tilde{k}, s_l\} = \max_{0 \le k' \le k} \left\{ t_{l-1,k-k'} + l \times \min\{k', s_l\} \right\} . \quad \square$$

We need the same dynamic programming algorithm for each $D_{(i,j)}(\mathcal{S})$, and we will denote the variables in Eq. (28) by $t^{i,j}_{l,k}$.

In the Erdős–Gallai inequalities, we also need to compute $\sum_{h=k+1}^{n} \min\{k, d_h\}$. We can define the degree sequence $\tilde{D}$ as $\tilde{d}_h := \min\{k, d_h\}$, and then

$$\sum_{h=k+1}^{n} \min\{k, d_h\} = \sum_{g=1}^{n} \tilde{d}_g - \tilde{t}_k$$

where $\tilde{t}_k$ is the sum of the k largest degrees in $\tilde{D}$. This can be obtained by the dynamic programming recursion

$$\tilde{t}_{l,0} = 0 \tag{29}$$

$$\tilde{t}_{l,k} = \max_{0 \le k' \le k} \left\{ \tilde{t}_{l-1,k-k'} + \min\{l, k\} \times \min\{k', s_l\} \right\} \quad \forall\, l > 1, \tag{30}$$

and then defining $\tilde{t}_k := \tilde{t}_{i,k}$ The correctness of Eqs. (29) and (30) can be proved similarly to the proof of Lemma 20. We need the same dynamic programming algorithm for each $D_{(i,i)}(\mathcal{S})$, and we will denote the variables in Eq. (30) by $\tilde{t}^{i}_{l,k}$.

Now we are ready to prove the theorem on simple graph realizations.

Theorem 21. *Let $D = (d_1, d_2, \ldots, d_n)$ and $F = (f_1, f_2, \ldots, f_n)$ be a pair of degree and neighbor degree sequences. Let Δ denote the largest degree in D. Then (D, F) is simple graph realizable if and only if the integer programming feasibility problem $\{\mathcal{X}, \mathcal{L}\}$ has a solution, where $\mathcal{X}$ consists of all x_λ variables for all partitions $\lambda = [\lambda_1, \lambda_2, \ldots, \lambda_{h(\lambda)}]$ such that $h(\lambda) \le \Delta$ and for all i, $\lambda_i \le \Delta$ and auxiliary variables $p_1, p_2, \ldots, p_\Delta$, $t^{i,j}_{l,k}$ for all $1 \le i \le j \le \Delta$, $1 \le k \le \Delta - 1$ and $1 \le l \le \Delta$, $\tilde{t}^{i}_{l,k}$ for all $1 \le i \le \Delta$ and $1 \le k, l \le \Delta$, $z_1, z_2, \ldots, z_r$, $b_1, b_2, \ldots, b_r$, with $r = O(\Delta^5)$ and $\mathcal{L}$ consists of*

- *the inequality system given in Eqs.* (19) and (20),
- *for all* $1 \leq i \leq j \leq \Delta$, *the integer linear programming feasibility problem defining the dynamic programming recursions in Eqs.* (27) and (28), *and for all* i, *the integer linear programming feasibility problem defining the dynamic programming recursions in Eqs.* (29) and (30),
- *for all* $1 \leq i < j \leq \Delta$, *the equation*

$$\sum_{\lambda|h(\lambda)=i} x_\lambda N(\lambda, j) = \sum_{\lambda'|h(\lambda')=j} x_{\lambda'} N(\lambda', i), \tag{31}$$

as well as the linear inequality system for the first $\Delta - 1$ *Gale–Ryser inequalities presented as*

$$t^{i,j}_{i,k} \leq \sum_{\lambda|h(\lambda)=j} x_\lambda \min\{k, N(\lambda, i)\} \tag{32}$$

- *and for all* i, *the equation*

$$\sum_{\lambda|h(\lambda)=i} x_\lambda N(\lambda, i) = 2p_i. \tag{33}$$

as well as the first Δ *Erdős–Gallai inequalities expressed as*

$$t^i_{i,k} \leq k(k-1) + \left(\sum_{\lambda|h(\lambda)=i} x_\lambda \min\{k, N(\lambda, i)\} \right) - \tilde{t}^i_{i,k}. \tag{34}$$

Proof. Similar to the proof of Theorem 18. The auxiliary variables b_i and z_i appear in the integer programming descriptions of minimization and maximization problems in the dynamic programming recursions. There are $O(\Delta^2)$ pairs of (i, j), for each fixed such pairs, there are $O(\Delta^2)$ $t_{l,k}$ auxiliary variables, that is, there are $O(\Delta^4)$ auxiliary variables to compute $t^{i,j}_{i,k}$'s. Similarly, it is easy to see that there are $O(\Delta^3)$ auxiliary variables to compute the auxiliary variables $t^i_{i,k}$ and $\tilde{t}^i_{i,k}$. For each of them, there are $O(\Delta)$ minimization and maximization problems in Eqs. (28) and (30). Thus, there are $O(\Delta^5)$ auxiliary variables of b's and z's. □

The theorem for forest realizations does not need inequalities with minimums and/or maximums.

Theorem 22. *Let* $D = (d_1, d_2, \ldots, d_n)$ *and* $F = (f_1, f_2, \ldots, f_n)$ *be a pair of degree and neighbor degree sequences. Let* Δ *denote the largest degree in* D. *Then* (D, F) *is forest realizable if and only if the integer programming feasibility problem* $\{\mathcal{X}, \mathcal{L}\}$ *has a solution, where* $\mathcal{X}$ *consists of all* x_λ *variables for all partitions* $\lambda = [\lambda_1, \lambda_2, \ldots, \lambda_{h(\lambda)}]$ *such that* $h(\lambda) \leq \Delta$ *and for all* i, $\lambda_i \leq \Delta$ *and auxiliary variables* $p_1, p_2, \ldots, p_\Delta$ *and* $\mathcal{L}$ *consists of*

- *the inequality system given in Eqs.* (19) and (20),
- *for all* $1 \leq i < j \leq \Delta$, *the equation*

$$\sum_{\lambda|h(\lambda)=i} x_\lambda N(\lambda, j) = \sum_{\lambda'|h(\lambda')=j} x_{\lambda'} N(\lambda', i), \tag{35}$$

- *for all* i, *the equation*

$$\sum_{\lambda|h(\lambda)=i} x_\lambda N(\lambda, i) = 2p_i. \tag{36}$$

- *and for all subsets* $\mathcal{I} \subseteq \{(i, j)|1 \leq i \leq j \leq \Delta\}$, *the inequality*

$$\sum_{(i,j)\in\mathcal{I}|i\neq j} \left(\sum_{\lambda|h(\lambda)=i} x_\lambda N(\lambda, j) + \sum_{\lambda|h(\lambda)=j} x_\lambda N(\lambda, i) \right) + \sum_{(i,i)\in\mathcal{I}} \sum_{\lambda|h(\lambda)=i} x_\lambda N(\lambda, i) \leq$$
$$2 \sum_{\lambda} x_\lambda \min \left\{ 1, \sum_{j|(h(\lambda),j)\in\mathcal{I}} N(\lambda, j) \right\} - 2. \tag{37}$$

Furthermore, (D, F) *has a caterpillar realization if none of the degrees is* 0, $\sum_{i=1}^n d_i = 2n - 2$ *(which can be given by requiring that Eq.* (37) *holds with equality) and a solution exists with*

$$x_\lambda = 0 \tag{38}$$

for all λ *for which* $h(\lambda) > 1$ *and* $\sum_{j>1} N(\lambda, j) \leq 2$.

Proof. We show that the conditions are necessary and sufficient for the existence of a sequence of labeled stub-stars S that (according to Theorem 15) is multigraph realizable, and for all subsets of colors I, $D_I(S)$ is forest realizable. Note that the multigraph realizability is ensured by first three points of the statement of the theorem. The inequality in Eq. (37) expresses that the sum of the non-zero degrees $D_I(S)$ is less than twice the number of non-zero degrees minus 2. Together with the fact that the sum of the degrees is even (Eq. (36)), these are the necessary and sufficient conditions for forest-realizability according to Theorem 12.

For a caterpillar realization, we need the conditions from Theorem 16 to be fulfilled, that is, none of the degrees is 0, the sum of the degrees is $2n-2$ and each stub has at most 2 labels larger than or equal to 2. □

Note that the forest realization will be a tree realization if $\sum_{i=1}^{n} d_i = 2n-2$.

For each case, we transformed the realization problem into an integer programming feasibility problem. Both the number of equations and the number of variables depends on Δ and not on n. Therefore, if Δ is fixed, then the computational time to solve the integer programming feasibility problem grows only as a poly-log function of n due to the complexity of performing the arithmetic operations. The practicality of the approach is demonstrated in the next section.

5. Simulation results

In order to demonstrate the practical implications of the presented results, we devised a set of experiments which show the potential of our approach in certain types of applications.

Furthermore, we were also able to show where the limitations of this approach were. Since the original motivation for this work comes from chemistry, we considered evaluating a reconstruction of saturated hydrocarbons.

The first part of the task was to find feasible sequences of labeled stub-stars for which we construct an integer linear program. Even though, in general, this is a hard problem, the restricted integer programs that we use have a constant number of variables, which makes it solvable in polynomial time [14].

We wanted to see two main results (1) how fast we can find one solution (i.e., one tree from a given input sequence) and (2) how fast the number of solutions grows with the size of the input. We constructed two different testsets for each of these two goals. For the first goal we generated trees of sizes 100 to 1000 with a step of 100. For the second goal we generated testsets of sizes only from 10 to 80 with a step of 10 (for reasons that will be obvious from the results). For all sizes we generated 100 trees and the obtained results are all averages (average times and average counts) for these 100 measurements.

Each tree is generated with a simple recursive procedure described below. The procedure generates a random tree with maximal degree k such that for the current vertex,

1. it generates a random degree d from $[1 \dots k-1]$ (except for the first node where the interval is $[1 \dots k]$) uniformly at random,
2. it chooses a random partition of $n-1$ into d pieces, i.e., choose how large each of the d subtrees will be,
3. then it recursively repeats the process for all d neighbors until reaching the leaves.

In order to harness the best practices in integer programming, we chose a state-of-the-art solver and used it to construct feasible sequences of labeled stub-stars. Our goal was to use out-of-the-box IP solvers and evaluate its performance by varying the size of the problem. Since in chemical applications the input represents a feasible tree (i.e. we know there exists a feasible solution) we also start with a tree, decompose it into a degree sequence and a sequence of sums of neighbor degrees.

We used an openly accessible SCIP solver [9] and in order to generate a specific integer program we used the ZIMPL language [13]. The final trees were reconstructed with the custom written program in Python.

The main two results are presented in Figs. 2 and 3. Fig. 2 shows that the time required to find one feasible solution does not increase in the range that we tested it. It is not surprising because neither the number of variables nor the number of inequalities in the integer programming depends on the input size. In the indicated range of input size, most of the time is spent on solving the integer programming problem and only a minor part of the running time of the solver is spent on actually building the tree.

The algorithm for creating a tree from a sequence of labeled stub-stars (the output of the integer programming solver) is the following: We first create a multigraph realization of the sequence of labeled stub-stars G. Then while G is not minimal among all realizations in the number of components, we apply the procedure described in the proof of Theorem 15 to produce a G' with one fewer component than G. By applying Depth-First Search algorithm to find cycles and paths in the procedure described in the proof of Theorem 15, it is easy to see that we can construct a tree in $O(|V|^2|E|^2 + |V||E|^3)$. In practice, we can construct an initial multigraph solution by packing monochromatic forests together. This good start causes the running time of the construction algorithm in practice is less than the time spent on the integer programming part.

Fig. 3 shows the exponential growth of the average number of feasible solutions to the given input. Already with $n = 30$ carbon atoms, the average number of solutions to a random input approaches 10 and with $n = 40$ carbon atoms, the average number of solutions is about 86. The average number of solutions keeps growing exponentially with the

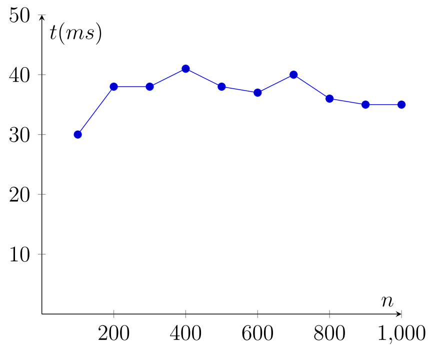


Fig. 2. The average time (t, measured in milliseconds) needed to generate one feasible solution of a tree with n vertices, maximum degree 4 and prescribed degree and sum of neighbor degree sequences. See text for details about generating input trees and finding a solution.

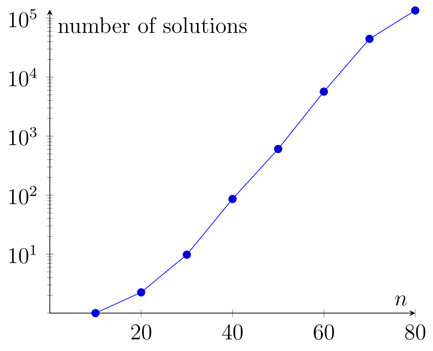


Fig. 3. The average number of solutions of a (degree, sum of neighbor degree) sequence of length n. See text for detail.

number of carbon atoms, indicating that it is not feasible to reconstruct large hydrocarbons using only NMR data and then testing each potential solution.

In chemistry however, besides the information about the neighborhood, other information could be used. Some of those chemical and physical limitations also infer various structural limitations which need to be further explored. Namely, completely unrestricted trees yield chemically unfeasible compounds so further work needs to be focused on trees with special structures so that the exponential growth of the number of solutions is postponed even further thus making this approach practical for even larger compounds.

6. Discussion

In this paper, we considered the problem of constructing a bounded degree graph with prescribed degree and neighbor degree sequence. The key is to find an appropriate sequence of labeled stub-stars that can realize the given degree and neighbor degree sequence. We talked about monochromatic subgraphs that considers only stubs with given labels incident to vertices with given degrees. This approach highly resembles the color-degree matrix problem that is also intensively studied [1,2,11,12]. The input of a color-degree matrix problem is a $k \times n$ matrix M, and the output is an edge-colored simple graph $G = (V, E)$, $|V| = n$. The number of colors is k, and for each color c_i, the degree sequence of the monochromatic subgraph of G for that color is the ith row of M.

Indeed, a sequence of labeled stub-stars, $\mathcal{S} = (S_1, S_2, \dots, S_n)$, can be represented by a *special color-degree matrix* M. M has $\binom{\Delta}{2} + \Delta$ rows and n columns, where Δ is the maximum degree in $\mathcal{S}$. The rows are labeled by $(i,j) \in \{(i,j) | 1 \leq i \leq j \leq \Delta\}$. When (i,j) is the label of the kth row, then

$$m_{k,l} = \begin{cases} \text{number of } j \text{ labels of } S_l & \text{if } S_l \text{ has degree } i \\ \text{number of } i \text{ labels of } S_l & \text{if } S_l \text{ has degree } j \\ 0 & \text{otherwise} \end{cases}$$

We call the (i,j) labels *colors*. We will index the rows of M by their color indexes, that is, we will talk about the (i,j)th row and we index entries as $m_{(i,j),l}$.

Color-degree matrices appeared earlier in the scientific literature, see for example [12]. The color-degree matrices introduced here are special in two ways: first, for any two rows with indexes (i,j) and (i',j') such that $i \notin \{i',j'\}$ and

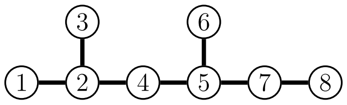


Fig. 4. Caterpillar counterexample C_1. See text for details.

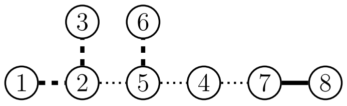


Fig. 5. Caterpillar counterexample C_2.

$j \notin \{i', j'\}$ and for any column index l, $m_{(i,j),l} \neq 0$ implies that $m_{(i',j'),l} = 0$. Second, we consider only special realizations of these special color-degree matrices.

Indeed, observe that not all edge disjoint monochromatic realizations of the degree sequences in the rows of M are realizations of the corresponding sequence of labeled stub-stars. For each $i < j$, it is required that the monochromatic realization be a *forced* bipartite realization, that is, we require that each edge goes between prescribed vertex classes.

To see why this is necessary, consider the caterpillar C_1, shown in Fig. 4. Its color-degree matrix will have the form

$$M = \begin{array}{c} (1,2) \\ (1,3) \\ (2,3) \\ (1,1) \\ (2,2) \\ (3,3) \end{array} \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 2 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 2 & 2 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Notice that the plain, dashed, dotted subgraphs in Fig. 5, respectively, are edge disjoint realizations for the $(1, 2)$-th, $(1, 3)$-th, and $(2, 3)$-th rows of M, respectively. However, their union is C_2 which is not a realization of M because C_2 has $(2, 2)$ and $(3, 3)$ edges. Also observe that the dotted subgraph in Fig. 5 is a bipartite graph, however, it is not a forced bipartite graph.

In general, it is NP-complete to decide if a color-degree matrix has any realization [5,10]. On the other hand, Carroll and Isaak showed that the problem is easy if for each subset of rows, the corresponding column sums form a degree sequence of a forest [3]. Hillebrand and McDiarmid showed that the problem remains easy if the column sums form a degree sequence that has a realization with at most one cycle [12]. These results cannot be directly applied to our problem since we require special realizations as discussed above.

Nonetheless, it is worth mentioning that the hard part of the degree and neighbor degree realization problem is to find the appropriate sequence of labeled stub-stars. Indeed, it remains easy to decide if a special color-degree matrix has a simple graph realization even if there is no bound on the maximal degree: due to the speciality of the color-degree matrices obtained from sequence of labeled stub-stars, it is guaranteed that the union of monochromatic simple graph realizations will remain simple graphs (given that the realizations for (i, j) colors, $i < j$ are forced bipartite).

Data availability

No data was used for the research described in the article.

Acknowledgments

I.M. was supported by NKFIH, Hungary grants KH126853, K132696 and SNN135643. The project is a continuation of the work done at the 2020 Budapest Semesters in Mathematics. All authors would like to thank the BSM for running the program.

References

[1] C. Bentz, M.-C. Costa, C. Picouleau, B. Ries, D. de Werra, Degree-constrained edge partitioning in graphs arising from discrete tomography, J. Graph Algorithms Appl. 13 (2) (2009).
[2] A. Busch, M. Ferrara, S. Hartke, M. Jacobson, H. Kaul, D. West, Packing of graphic n-tuples, J. Graph Theory 70 (1) (2012).
[3] J. Carroll, G. Isaak, Degree matrices realized by edge-colored forests, 2009, Online; http://www.lehigh.edu/~gi02/ecforest.pdf.
[4] É. Czabarka, A. Dutle, P.L. Erdős, I. Miklós, On realizations of a joint degree matrix, Discrete Appl. Math. 181 (30) (2014) 283–288.
[5] C. Dürr, F. Guíñez, M. Matamala, Reconstructing 3-colored grids from horizontal and vertical projections is NP-hard: A solution to the 2-atom problem in discrete tomography, SIAM J. Discrete Math. 26 (1) (2012).
[6] P. Erdős, T. Gallai, Graphs with vertices of prescribed degrees, Mat. Lapok 11 (1960) 264–274 (in Hungarian).
[7] E.L. Erdős, I. Miklós, Not all simple looking degree sequence problems are easy, J. Comb. 9 (3) (2018) 553–566.

[8] D. Gale, A theorem on flows in networks, Pacific J. Math. 7 (2) (1957) 1073–1082.
[9] G. Gamrath, D. Anderson, K. Bestuzheva, W. Chen, L. Eifler, M. Gasse, P. Gemander, A. Gleixner, L. Gottwald, K. Halbig, G. Hendel, C. Hojny, T. Koch, P. Le Bodic, S.J. Maher, F. Matter, M. Miltenberger, E. Mühmer, B. Müller, M.E. Pfetsch, F. Schlösser, F. Serrano, Y. Shinano, C. Tawfik, S. Vigerske, F. Wegscheider, D. Weninger, J. Witzig, The SCIP Optimization Suite 7.0, ZIB-report, Zuse Institute Berlin, 2020, pp. 10–20.
[10] R. Gardner, P. Gritzmann, D. Prangenberg, On the computational complexity of reconstructing lattice sets from their X-rays, Discrete Math. 202 (1–3) (1999) 45–71.
[11] F. Guíñez, M. Matamala, S. Thomassé, Realizing disjoint degree sequences of span at most two: A tractable discrete tomography problem, Discrete Appl. Math. 159 (1) (2011) 23–30.
[12] A. Hillebrand, C. McDiarmid, Colour degree matrices of graphs with at most one cycle, Discrete Appl. Math. 209 (2016) 144–152.
[13] Thorsten Koch, Rapid Mathematical Programming (Ph.D. thesis), 2005.
[14] Hendrik W. Lenstra, Integer programming with a fixed number of variables, Math. Oper. Res. 8 (1983) 538–548.
[15] C. Orsini, M. Dankulov, P. Colomer-de Simón, et al., Quantifying randomness in real networks, Nature Commun. 6 (8627) (2015).
[16] H.J. Ryser, Combinatorial properties of matrices of zeros and ones, Canad. J. Math. 9 (1957) 371–377.
[17] I. Stanton, A. Pinar, Constructing and sampling graphs with a prescribed joint degree distribution, ACM J. Exp. Algorithmics 17 (3) (2012) 5.