Degree-based connected graph construction with assortativity constraint

Ke, Yingyue; Van Mieghem, Piet

DOI
10.1088/1367-2630/ae09d4

Publication date
2025

Document Version
Final published version

Published in
New Journal of Physics

New Journal of Physics
The open access journal at the forefront of physics

PAPER • OPEN ACCESS

Degree-based connected graph construction with assortativity constraint

To cite this article: Yingyue Ke and Piet Van Mieghem 2025 *New J. Phys.* **27** 103901

IOP Publishing New J. Phys. 27 (2025) 103901 https://doi.org/10.1088/1367-2630/ae09d4

New Journal of Physics
The open access journal at the forefront of physics

Deutsche Physikalische Gesellschaft DPG
IOP Institute of Physics

Published in partnership with: Deutsche Physikalische Gesellschaft and the Institute of Physics

PAPER

Degree-based connected graph construction with assortativity constraint

Yingyue Ke* and **Piet Van Mieghem**

Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, 2628 CD Delft, The Netherlands
* Author to whom any correspondence should be addressed.

E-mail: y.y.ke@tudelft.nl

Keywords: graph construction, degree sequence, connected graphs, assortativity

OPEN ACCESS

RECEIVED
8 May 2025

REVISED
15 September 2025

ACCEPTED FOR PUBLICATION
22 September 2025

PUBLISHED
30 September 2025

Abstract

Degree-based graph construction is a fundamental problem in network science. A graph is simple if there are no self-loops and no multiple links between any pair of nodes in the graph. A degree sequence $d = (d_1, d_2, \cdots, d_N)$ is graphical if d can be represented as the degree sequence of at least one simple graph, where the graph is called a realization of the sequence d. In this work, we introduce a novel method (LSFGR) for generating simple graphs from graphical degree sequences, focusing additionally on connectedness and on assortativity. LSFGR guarantees connected graphs for all potentially connected degree sequences. In the case where a degree sequence has no simple realization, LSFGR produces graphs with at most one node with self-loops. In addition, the graphs generated from LSFGR characterize real-world networks with medium assortativity.

1. Introduction

Network models are powerful tools for representing and analyzing a wide range of empirical systems, including computer, social, biological, transportation, and telecommunication networks. By representing real-world systems as graphs, where components of the systems are represented by the nodes and the interactions among the components are represented as the links in the graphs, we can capture complex relationships and interactions between components [1]. Various graph metrics are used to analyze networks and to gain insights into their topology and function [2]. For example, the clustering coefficient measures how interconnected a node's neighbors are. Assortativity indicates the tendency of nodes to connect to other nodes with similar degree. Betweenness centrality quantifies how often a node lies on shortest paths between other nodes. In many practical scenarios, however, complete knowledge of the network's structure is unavailable. Some structural metrics may be known, but the exact connections between nodes remain hidden. In such cases, reconstructing the underlying graph from partial or indirect data is non-trivial. For instance, one might reconstruct the network topology from a demand matrix in telecommunication systems [3], or infer the graph from subgraph samples when large-scale data collection is limited [4].

One particularly challenging problem in network construction is generating graphs with given degree sequences. A network is represented by a graph $G(N, L)$ that consists of N nodes and L links. Each link connects two nodes. The 0-1 adjacency matrix $A = (a_{ij})$ of order N indicates a link between nodes i and j if $a_{ij} = 1$. Graphs without self-loops and multiple links are called simple, i.e. any two nodes have at most one link between them and no node is connected to itself. Otherwise, a graph is non-simple. The degree sequence of a graph $G(N, L)$ is defined as $d = (d_1, d_2, \cdots, d_N)$, where degree d_i is the number of nodes that connect to the node i. If the nodes are labeled appropriately, then the sequence $d = (d_{(1)}, d_{(2)}, \cdots, d_{(N)})$ is nonincreasing, where $d_{(k)}$ represents the k-largest degree. We use $d = (d_1, d_2, \cdots, d_N)$ to represent a nonincreasing degree sequence for convenience in this paper. The degree-based graph construction problem involves determining whether a given degree sequence d can represent the degree sequence of a simple graph. We call a degree sequence $d = (d_1, d_2, \cdots, d_N)$ graphical if the sequence d can be represented as the degree sequence of a simple graph G. A walk from node i to node j is a succession of k links $(r_0 \sim r_1)(r_1 \sim r_2)\cdots(r_{k-1} \sim r_k)$, where the node label $r_1 = i$ and $r_k = j$. A path is a walk in which all nodes

are different. A graph is connected if and only if there is a walk between any two nodes i and j in the graph. A degree sequence is potentially connected if there exists a connected graph that realizes the sequence.

The degree-based graph construction problem is typically categorized into two main classes: (a) constructing a simple graph that satisfies both the given degree sequence and additional structural constraints. (b) constructing all simple graphs with a given degree sequence. The incorporation of structural constraints is motivated by practical applications. For example, many real-world networks such as social, communication, biological, and transportation systems, require connectivity to function effectively. Therefore, verifying the existence of a connected realization of a degree sequence is critical [5]. Determining whether a given sequence is graphical is addressed by the Erdös-Gallai Theorem [6] and Havel-Hakimi Theorem [7, 8], which give the necessary and sufficient conditions. Applying the Havel-Hakimi Theorem to generate a simple graph for a graphical degree sequence is discussed in detail in section 2. The authors [9] prove that there is a connected realization of a degree sequence $d = (d_1, d_2, \cdots, d_N)$ of a graph (not necessarily simple) if and only if $\frac{1}{2}\sum_{i=1}^{N} d_i \geqslant N-1$ and $d_i \neq 0, \forall i$ or if $N = 1$. Additional structural constraints such as degree correlations, which capture the tendency of nodes to connect based on similarity or dissimilarity of degree, are also important in modeling real networks [10]. These constraints enable more accurate representations of empirical network structures and dynamics.

The existing approaches for sampling all simple graphs with a given degree sequence generally involve the stub matching–configuration model [1], link rewiring–Markov chain Monte Carlo (MCMC) methods [11] and greedy algorithms [12]. The basic idea of the configuration model is: given a degree sequence, each node is assigned a number of stubs equal to its degree. Pairs of stubs are then randomly selected and connected to form full edges. The process is repeated until no unconnected stubs remain. However, the method may produce graphs that are not simple but contain self-loops or multiple edges. In such cases, the generated graph is discarded, and the process is restarted from the beginning. Nevertheless, the configuration model remains a cornerstone of network modeling due to its conceptual simplicity. MCMC typically begins with an initial valid simple graph that realizes the given degree sequence and then applies link swapping operations, which preserve the degrees of all nodes and do not introduce self-loops or multiple links [13]. Two links between nodes $i \sim j$ and $m \sim n$ can be rewired to $i \sim n$ and $m \sim j$. Repeating such link rewiring leads to sampling over the space of all simple graphs with the same given degree sequence. MCMC may mix slowly and take a large number of iterations before the samples approximate a uniform distribution over the desired graph space. Greedy algorithms are based on the star-constrained graphicality theorems [12, 14]. Their statistical weights are computable and can be used to estimate properties that characterize the ensemble of realizations of a degree sequence.

This paper addresses the problem of constructing a simple, connected graph with a given degree sequence and a specified level of assortativity. We assume that all degree sequences discussed in this work are graphical unless mentioned otherwise. Our goal is to generate connected and simple graphs exhibiting moderate assortativity for degree sequences that are potentially connected. We propose an algorithm called the Largest-Smallest-First Graph Realization (LSFGR), which produces a connected graph for any potentially connected degree sequence. For graphical degree sequences, we provide conditions ensuring that LSFGR returns a simple graph. Our results demonstrate that LSFGR generates graphs with moderate assortativity, complementing the Havel-Hakimi algorithm.

The remainder of the paper is outlined as follows. We review and provide a detailed discussion about the Havel-Hakimi algorithm in section 2. Section 3 presents our novel graph realization method. We analyze the mechanisms and provide examples. In section 4, we investigate the performance of our algorithm on connectedness, assortativity, modularity and clustering coefficient. We conclude our results in section 5.

2. Havel-Hakimi algorithm preview

2.1. Connectedness

We consider degree sequence realization algorithms with the following structure:

Step 1: Choose a node i to be the 'hub' node.

Step 2: Connect the hub node i to other nodes until it has degree d_i.

Step 3: Repeat steps 1 and 2 until all nodes have the degree specified by the degree sequence d.

Different algorithms have different methods to choose the hub in step 1 and to choose the neighbors of the hub in step 2. We define three different degrees when applying an algorithm to generate a graph for a degrees sequence $d = (d_1 = s, d_2, d_3, \cdots, d_N)$. The degree d_i is called the final degree of a node i. The intermediate degree d_i' is defined as the degree of node i during the graph construction process. The remaining degree $t_i = d_i - d_i'$ is the difference between the final and intermediate degree of node i. The construction process is complete and returns a graph when the intermediate degrees d_i' reach the final degrees d_i for all nodes, i.e. the remaining degrees t_i reach zero.

The Havel-Hakimi theorem is the most foundational result in graph theory to check if a degree sequence d is graphical and can be applied to realize a sequence d, which states that if a nonnegative sequence $d=(d_1=s,d_2,d_3,\cdots,d_N)$ is graphical if only if the sequence $(d_2-1,d_3-1,\cdots,d_{s+1}-1,\cdots,d_N)$ is graphical. Applying the Havel-Hakimi Theorem to generate a graph consists of the following steps:

1. Start from an empty graph G with N nodes and a degree sequence $d'=(d'_1=0,\cdots,d'_N=0)$.
2. Choose an arbitrary node i with remaining degree $t_i>0$ as the hub.
3. Connect the hub i to the t_i nodes whose indexes are the indexes of the first t_i largest elements in the remaining degree sequence t except the element t_i itself.
4. Repeat steps 2 and 3 until $t=(0,0,\cdots,0)$.

In the Havel-Hakimi algorithms, there is no restriction on the choice of hub node, which can be selected arbitrarily. The key step is step (2), where the hub with the remaining degree t_i is connected to the other t_i nodes with the largest remaining degrees. Thus, the Havel-Hakimi algorithm constitutes a class of algorithms. Depending on the manner to choose a hub, the Havel-Hakimi algorithm can generate different realizations of a given degree sequence. A common idea is to choose the node with the largest remaining degree as the hub in each step, thereby prioritizing nodes with larger remaining degrees. We call this variant Largest-First Havel-Hakimi algorithm (LFHH). In LFHH, both the hub and the nodes linked to the hub have large remaining degrees, which indicates that LFHH tends to connect high-degree nodes to high-degree nodes as well as low-degree nodes to low-degree nodes. We show that LFHH does not guarantee to return a connected realization, given that the degree sequence is potentially connected.

LFHH cannot generate certain trees. A tree is a connected graph with N nodes and $N-1$ links in which there is no closed path (circle). In a tree, a node with degree 1 is called a leaf and a node with a degree larger than one is called a parent. The nodes connecting to the leaves are called support nodes. The number of leaves is the upper bound of the number of support nodes in a tree, because more than one leaf can connect to the same support node. All support nodes are parents. We show that LFHH cannot generate a tree of $N\geqslant 5$ nodes and $M\leqslant N-3$ leaves. To prove this result, we first establish the following lemma.

Lemma 1. *A tree with $N\geqslant 5$ nodes and $M\leqslant N-3$ leaves must have two leaves whose parents are not linked.*

Proof. A tree of $N\geqslant 5$ nodes has at least 2 leaves. When $M\leqslant N-3$, the tree has at least $N-M\geqslant 3$ parents. Two leaves i and j linked with two different support nodes p_i and p_j must exist, otherwise, all leaves have the same support node and the number of leaves would be $N-1$. Thus, the number of support nodes S is no less than two given a tree as defined in Lemma 1. We discuss two cases based on the value of S: (1) $S=2$; (2) $S\geqslant 3$.

(1) $S=2$. The two support nodes are defined as p_i and p_j. We assume that the link $p_i\sim p_j$ exists. All M leaves are linked with p_i or p_j. There is at least one node that still has to be added to the graph because $N\geqslant M+3$. Each node that still has to be added can not link with p_i or p_j, otherwise, there are more than M leaves. They can also not link with a leaf, because then the leaf, or one of its children, becomes a third support node, which contradicts $S=2$. Therefore, our assumption that the link $p_i\sim p_j$ exists can not hold.

(2)$S\geqslant 3$. The case of no less than 3 support nodes is straightforward. If all pairs of support nodes have a link, then the tree has a complete subgraph of no less than 3 nodes. Thus, the tree contains circles, which contradicts the definition of a tree. Therefore, at least two support nodes are not linked. □

Theorem 1. *Let $d=(d_1,d_2,\cdots,d_N)$ be a graphical degree sequence with $N\geqslant 5$ and suppose M entries of d are equal to one. If $\sum_{i=1}^{N}d_i=2N-2$, then LFHH generates (1) a tree if $M=N-1$ or $M=N-2$; (2) a disconnected graph G if $M\leqslant N-3$.*

Proof. The defined degree sequence represents the degree sequence of a tree of $N\geqslant 5$ nodes and M leaves. We discuss three different cases.

1 $M=N-1$. The only realization for $M=N-1$ is a star. Thus, LFHH generates a connected graph in which node 1 links to all remaining $N-1$ nodes.

2 $M=N-2$. The degree sequence has the form $d=\{d_1,d_2,\underbrace{1,1,\cdots,1}_{N-2}\}$. The sum of degrees of nodes 1 and 2 is $d_1+d_2=(2N-2)-(N-2)=N$. Applying LFHH, the generated graph is a tree, where node 1 connects to nodes $2,3,\cdots d_1+1$ and node 2 connects to nodes $1,d_1+2,d_1+3,\cdots,N$.

3 $M\leqslant N-3$. Using LFHH, a node with a higher remaining degree has a higher priority to be connected to a hub. Among all nodes with the same remaining degree, a node with a smaller label is linked to a hub before another node with a larger label. Nodes $N-1$ and N have the largest two labels and the smallest degree $d_{N-1}=d_N=1$. If we prove that the subsequence $\underline{d}:=(d_1,d_2,\cdots,d_{N-2})$ is graphical, then node $N-1$ will be the hub and connect to node N after $\underline{d}$ is realized. As a result, the nodes $N-1$ and N form

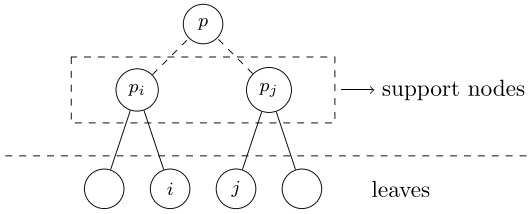


Figure 1. An illustration of the tree T. The leaves are below the dashed line and the parents are above the dashed line. The support nodes p_i and p_j of leaves i and j are in the dashed rectangle. Node p represents another node.

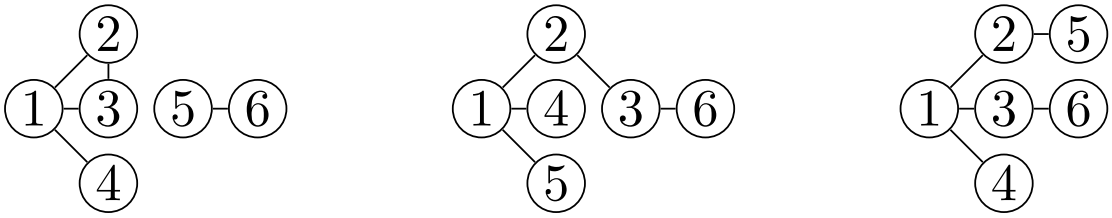


Figure 2. The realizations of the degree sequence $d = (3, 2, 2, 1, 1, 1)$ from LFHH(left), SFHH(middle) and LSFGR(right).

an individual component and do not connect to the other $N-2$ nodes in the graph G, implying that the graph G contain at least two components $\{1, 2, \cdots, N-2\}$ and $\{N-1, N\}$. The degree sequence d defined in Theorem 1 represents the degree sequence of a tree of N nodes. Thus, there exists a tree T that realizes d. Lemma 1 tells that there are two leaves i and j from different support nodes p_i and p_j that are not linked in T, as shown in figure 1. By removing two leaves i and j and connecting their support nodes p_i and p_j, we obtain the graph $\underline{T}$ with the degree sequence $\underline{d}$. Therefore, we prove that the degree sequence $\underline{d}$ is graphical.

□

In order to improve LFHH to generate connected graphs, Horvát and Modes have shown that picking the node with the smallest remaining degree as the hub guarantees connected realizations, given that the degree sequences are potentially connected [5]. We call this variant the Smallest-First Havel-Hakimi algorithm (SFHH). SFHH tends to connect high-degree nodes to low-degree nodes (the hubs), thereby generating connected graphs for potentially connected degree sequences. Considering the degree sequence $d = (3, 2, 2, 1, 1, 1)$, LFHH produces a disconnected graph, while SFHH produces a connected graph, as shown in figure 2.

To summarize, the Havel-Hakimi algorithm generates simple graphs for graphical degree sequences. LFHH tends to connect high-degree nodes with other high-degree nodes, which may not guarantee connectivity for potentially connected degree sequences. In contrast, SFHH connects high-degree nodes with low-degree nodes, ensuring the generation of connected graphs if the degree sequence is potentially connected. For other variants of the Havel-Hakimi algorithm, analyzing the performance of assortativity of the resulting realizations is intractable because of the arbitrariness of hubs.

2.2. Assortativity

As we have introduced in section 1, various metrics are applied to describe the structure and behavior of networks, such as degree distribution, centrality, assortativity and modularity. Among these, assortativity measures the correlation between connected nodes in a network [15], which helps to assess the resilience of a network [16], to provide insight into the spread of information or diseases in networks [17] and to detect communities within networks [18]. Thus, we consider assortativity when building graph models to represent real-world systems. Introduced by Newman [19], the formula of assortativity is derived as

$$\rho_D(G) = 1 - \frac{\sum_{i\sim j}\left(d_i - d_j\right)^2}{\sum_{i=1}^{N} d_i^3 - \frac{1}{2L}\left(\sum_{i=1}^{N} d_i^2\right)^2} \tag{1}$$

where $L = \frac{\sum_{i=1}^{N} d_i}{2}$ is the number of link in a graph and $i \sim j$ represents a link connected nodes i and j [20]. Positive assortativity means that high-degree nodes tend to connect to other high-degree nodes, while

negative assortativity indicates that they tend to connect to low-degree nodes. Medium assortativity occurs when there is no obvious correlation between the degrees of connected nodes.

In LFHH, the hubs are the nodes with the largest remaining degree and connect to other nodes with the largest remaining degrees, which implies that LFHH tends to connect high-degree nodes to high-degree nodes to generate graphs with high assortativity, such as in social networks [21], where people tend to associate with others in similar professional or social status. In contrast, SFHH tends to connect low-degree nodes to high-degree nodes to generate graphs with low assortativity. Thus, graphs generated by SFHH can characterize disassortative networks. For example, protein interaction networks typically show that a small number of highly connected proteins interact with a larger number of poorly connected proteins, thereby resulting in low assortativity [22]. However, some empirical networks have medium assortativity where high-degree nodes connect to both other high-degree nodes and low-degree nodes. Major airports connect to both other major and smaller regional airports in airline networks [23].

Based on the above discussion, we aim to design a novel algorithm that can generate a simple graph for every graphical degree sequence and exhibit a moderate level of assortativity, while ensuring that the graphs remain connected for potentially connected degree sequences. The approach is intended to model network systems with moderate assortative mixing, thereby addressing the limitations of the Havel-Hakimi algorithm in generating networks with medium assortativity.

3. Largest-smallest-first graph realization method

3.1. LSFGR algorithm

In this section, we propose the Largest-Smallest-First Graph Realization Algorithm (LSFGR) to generate connected graphs with medium assortativity for potentially connected degree sequences. We choose nodes as hubs based on final degrees. A node with a higher final degree is chosen before another node with a lower final degree. After a hub i is chosen, we connect it to the node j with the smallest intermediate degree d'_j. The next hub is considered when the current hub i has no remaining degree. To construct a graph with a degree sequence $d = (d_1, d_2, \cdots, d_N)$, LSFGR consists of the steps:

1. Start from an empty graph G with N nodes and a degree sequence $d' = (d'_1 = 0, \cdots, d'_N = 0)$.
2. Choose the node i from the nodes with remaining degree that has the largest final degree d_i as the hub.
3. Connect the hub i to the first node other than itself with the smallest intermediate degree.
4. Repeat step 3 until $d'_i = d_i$. If the hub i is the only node that has remaining degree, then self-loops are added.
5. Repeat steps 2-4 until $d' = d$.

Figure 3 presents the meta-code of LSFGR. We point out that multiple links are allowed in LSFGR. In addition, self-loops are added if the hub is the only node with a remaining degree; hence, the resulting graphs may not always be simple. However, we identify a necessary condition for producing simple realizations. We first detail the algorithm and then introduce a modified version in section 4.2 to ensure simplicity.

The computational complexity of LSFGR is $\mathcal{O}(Nd_{\max})$, where $d_{\max}$ represents the largest degree in d. The while-loop iterates through at most N hubs and at most $d_{\max}$ links per hub. Lines 5-11 in figure 3 can be executed in constant time. For lines 6-8,10 and 11, a constant time implementation is straightforward. The check in line 5 can be done in constant time by keeping track of the number of nodes that have a remaining degree of zero. The minimum in line 9 can be found in constant time after observing that given that the previously added link was from a hub to node j and the current hub is node i, the next node to link to the hub i must be node $j+1$ or node $i+1$. Thus, the LSFGR algorithms can be used to model empirical networks effectively.

3.2. LSFGR example

An example of applying LSFGR to construct a graph of 8 nodes for the sequence $d = (5, 4, 3, 3, 3, 3, 2, 1)$ is explained step by step in figure 4.

1. Start from an empty graph G with $N = 8$ nodes and with the degree sequence $d' = (0, 0, 0, 0, 0, 0, 0, 0)$.
2. Since $d'_1 < d_1$ and $0 = d'_j < d_j$ for $1 < j \leqslant N$, we choose node 1 as the hub and connect node 1 to next $d_1 - d'_1 = 5$ nodes, i.e. nodes 2, 3, 4, 5 and 6. As a result, the intermediate degree sequence becomes $d' = (5, 1, 1, 1, 1, 1, 0, 0)$.
3. Since $d'_2 < d_2$ and $0 = d'_{\min} = d'_7 < d_7$, node 2 is the hub and is connected to node 7. The degree sequence becomes $d' = (5, 2, 1, 1, 1, 1, 1, 0)$. Next, node 2 is connected to node 8 since $0 = d'_{\min} = d'_8 < d_8$. Now, the degree sequence becomes $d' = (5, 3, 1, 1, 1, 1, 1, 1)$ and $d'_{\min} = 1$. Since node 2 still has one remaining

LSFGR ALGORITHM

Input: A nonincreasing degree sequence $d = (d_1, d_2, \cdots, d_N)$
Output: A graph G **with the degree sequence** d

$G \leftarrow$ an empty graph with N nodes
$d' = (0, \cdots, 0) \leftarrow$ the degree sequence of G
$i \leftarrow 1$
while $i \leq N-1$ and $d'_i < d_i$ **do**
    if $d_k = d'_k = 0,\ \forall k > i$
        add link $(i,\ i)$ to graph G a total of $(d_i - d'_i)/2$ times
        end while
    end if
    $d'_j \leftarrow$ the first element in d' satisfying $d'_j = \min_{k>i, d'_k \neq d_k} d_k$
    add link (i, j) to graph G
    $d'_i \leftarrow d'_i + 1$, $d'_j \leftarrow d'_j + 1$, $i \leftarrow i + 1$
end while
return G

Figure 3. Meta code for LSFGR.

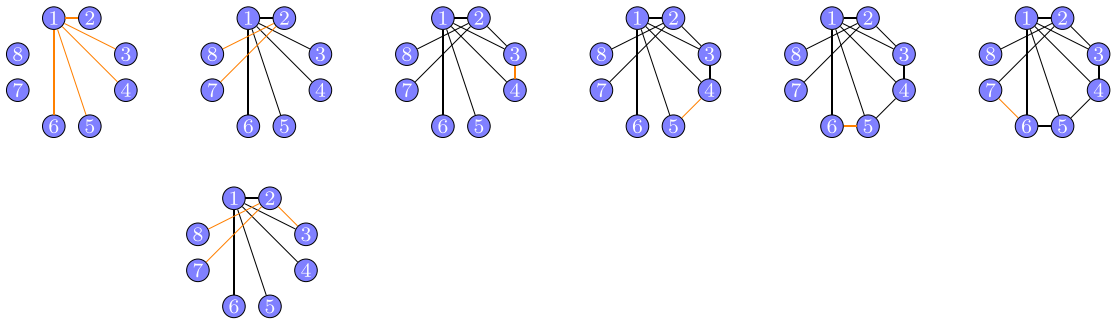


Figure 4. The process to generate a graph with a degree sequence $d = (5, 4, 3, 3, 3, 3, 2, 1)$ by LSFGR.

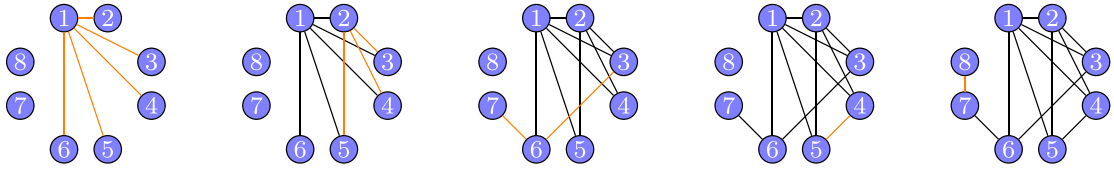


Figure 5. The process to generate a graph with a degree sequence $d = (5, 4, 3, 3, 3, 3, 2, 1)$ by LFHH.

degree, we continue to connect node 2 to node 3, which is the first node with a minimum degree $d'_{\min} = d'_3 < d_3$. Thus, we have $d' = (5, 4, 2, 1, 1, 1, 1, 1)$.

4. Since $d'_3 < d_3$ and $1 = d'_{\min} = d'_4 < d_4$, we choose node 3 as the hub and add link $3 \sim 4$. Thus, the degree sequence becomes $d' = (5, 4, 3, 2, 1, 1, 1, 1)$.
5. Since $d'_4 < d_4$ and $1 = d'_{\min} = d'_5 < d_5$, node 4 becomes the hub and is linked with node 5. Then, the degree sequence becomes $d' = (5, 4, 3, 3, 2, 1, 1, 1)$.
6. Since $d'_5 < d_5$ and $1 = d'_{\min} = d'_6 < d_6$, link $5 \sim 6$ is added. Therefore, the degree sequence becomes $d' = (5, 4, 3, 3, 3, 2, 1, 1)$.
7. We complete the construction by connecting node 6 to node 7 and obtain $d' = d$.

For the same degree sequence $d = (5, 4, 3, 3, 3, 3, 2, 1)$, the graph generated by LFHH is illustrated in figure 5. In LSFGR, the connected component of all 8 nodes forms earlier, when node 2 acts as the hub.

Whereas in LFHH, this connected component of all 8 nodes occurs later with node 7 as the hub. The observation suggests that LSFGR more actively integrates nodes during the graph construction process.

4. LSFGR performance

4.1. LSFGR connectedness

In section 3.1, we observe that LSFGR tends to include more nodes in the intermediate subgraph and prioritizes the generation of connected components with greater sizes. We show that LSFGR returns a connected graph if the given degree sequence is potentially connected.

Theorem 2. *LSFGR returns a connected graph for a potentially connected degree sequence* $d = (d_1, d_2, \cdots, d_N)$.

Proof. We define the graph G with nodes $\{1, 2, \cdots, N\}$, which is returned from LSFGR with the degree sequence $d = (d_1, d_2, \cdots, d_N)$ and we define $C_1 \subset G$ as the component of G that includes node 1. $|C_1| = K$ denotes the number of nodes in C_1. $d_1 + 1 \leqslant |C_1| = K \leqslant N$ because node 1 has d_i neighbors. We prove that the graph G is connected by showing $|C_1| = N$. The component C_1 can have two structures: (I) The component C_1 consists of the first K consecutively labelled nodes in graph G; (II) The nodes in C_1 are not labelled consecutively.

Case I: The nodes in C_1 are labelled consecutively and component C_1 consists of the first K nodes in graph G, i.e. $G = \{\underbrace{1, 2, \cdots, K}_{C_1}, K+1, \cdots, N\}$.

Since C_1 is connected, there exists one step where the first link of node K is added, which is the link $p \sim K$, as shown in figure 6. When link $p \sim K$ is added, the hub is node p, which means that all nodes $i < p$ already reach their final degree d_i. Now, we consider node p. We assume that node p does not reach its final degree d_p after the link $p \sim K$ is added. Since all nodes after node K do not have a link yet, node p will link with node $K+1$ in the next step. Thus, component C_1 will include node $K+1$ and then $|C_1| > K$, which contradicts our assumption. Therefore, the link $p \sim K$ is the last link added to C_1.

The degree sequence of C_1 is $d(C_1) = (d_1, d_2, \cdots, d_p, d_{p+1} = \cdots = d_K = 1)$, which indicates

$$\sum_{j=p+1}^{K} d_j = K - p. \tag{2}$$

All nodes j for $p+1 \leqslant j \leqslant K$ in C_1 have the same degree $d_j = 1$, because they have not been the hub yet.

Examining the structure of C_1 in figure 6, we obtain

$$d_1 + \sum_{i=2}^{p} (d_i - 1) = K - 1,$$

from which we find

$$\sum_{i=1}^{p} d_i = d_1 + \sum_{i=2}^{p} (d_i - 1) + (p-1) = K + p - 2. \tag{3}$$

Combining equations (2) and (3) results in $\sum_{i=1}^{K} d_i = 2K - 2$. Thus, the component C_1 is a connected graph of K nodes and $K-1$ links. Since the given degree sequence $d = (d_1, d_2, \cdots, d_N)$ is nonincreasing and potentially connected, it holds that $1 \leqslant d_j \leqslant d_K = 1$ for $K+1 \leqslant j \leqslant N$, indicating $d_j = 1$ for $K+1 \leqslant j \leqslant N$. We reach

$$\sum_{i=1}^{N} d_i = \sum_{i=1}^{K} d_i + \sum_{i=K+1}^{N} d_i = 2K - 2 + N - K = N + K - 2. \tag{4}$$

Moreover, a potentially connected sequence d implies $\sum_{i=1}^{N} d_i = N + K - 2 \geqslant 2N - 2$, therefore, $K \geqslant N$. Combining $K \leqslant N$ leads to $K = N$, which proves that the graph G is connected.

Case II: The nodes in C_1 are not labelled consecutively. We define the set of nodes in C_1 as $V(C_1) = \{V_1, V_2, \cdots, V_k\}$, where each V_j contains consecutively labelled nodes or one node for $1 \leqslant j \leqslant k$ and V_k contains at least one node label larger than K.

We denote one node that is between V_1 and V_2 and is not in C_1 as node x. Since component C_1 is connected, there exists one set $V_j, 2 \leqslant j \leqslant k$ that contains a node m connected to one node $n \in V_1$, where $n < m$. The first link of node m appears no later than link $n \sim m$. For any node $q < m$, its first link appears earlier than the first link of node m. Thus, all nodes q and specifically node x will link with a node in V_1, which contradicts that node x is not an element of C_1. □

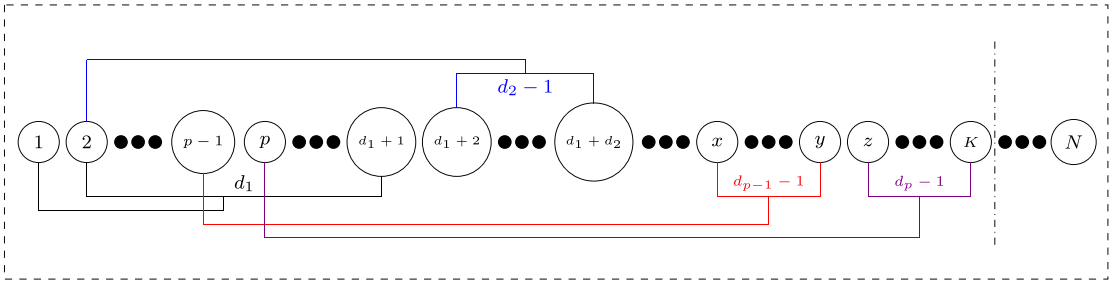


Figure 6. The whole picture represents the graph G. The left part of the vertical dashed line represents the component C_1. All hollow and solid dots represent nodes. Inside the hollow dots are the labels. Straight solid lines represent links between the hub and other nodes. The numbers above (below) the lines are the number of nodes linked to the hub. For example, node 1 is connected to d_1 nodes labelled as $\{2,\cdots,d_1+1\}$.

LSFGR outperforms LFHH in generating connected graphs. Both LSFGR and SFHH produce connected graphs for potentially connected degree sequences. However, realizations may differ up to isomorphism (see figure 2).

4.2. LSFGR simpleness

4.2.1. Theory

We have mentioned that the realization by LSFGR is not always simple for a graphical degree sequence in section 3.1. Both self-loops and multiple links can appear. The question is whether we can find where self-loops or multiple links appear in a graph generated by LSFGR. If yes, then we can adjust the algorithm to generate simple graphs. Actually, we do. We investigate a necessary condition to generate simple graphs with LSFGR and provide the modified LSFGR algorithm to ensure simple realizations for all graphical degree sequences in this section. A family of subsequences of a degree sequence d is defined before continuing the discussion.

Definition 1. Given a nonzero degree sequence $d=(d_1,d_2,\cdots,d_N)$ with $N\geqslant 2$ and $1\leqslant d_i\leqslant N-1$, the max-tree-subsequence $MTS(d)=(m_1,\cdots,m_k,m_{k+1},m_{k+2},\cdots,m_N)$ is defined as

$$MTS(d)=(m_1=d_1,\cdots,m_k=d_k,m_{k+1},m_{k+2}=\cdots=m_N=1)\,. \tag{5}$$

where $m_{k+1}:=2N-2-\sum_{i=1;\ i\neq k+1}^{N}m_i$ and k is the largest number that ensures $1\leqslant m_{k+1}\leqslant d_{k+1}$. If $MTS(d)$ exists, then we call d a MTS-sequence. The tree generated by LSFGR with degree sequence $MTS(d)$ is called the MTS-tree.

Definition 1 specifies the degree sequence of the spanning tree that maximizes the number of leaves for a degree sequence d. We define $d:=\delta_0$ and $\delta_{i+1}=\delta_i-MTS(\delta_i)$. Here, and in the following, addition and subtraction of sequences are both element-wise operations. Consider the degree sequence $\delta_0:=d=(5,4,3,3,3,3,2,1)$ discussed in section 3.2. The sequence $MTS(\delta_0)$ contains $N=8$ elements. We calculate $m_{k+1}=(2\times 8-2)-\sum_{i=1}^{k}d_i-\sum_{i=k+2}^{N=8}1=7+k-\sum_{i=1}^{k}d_i$. If $k=1$, then $m_{k+1}=m_2=3$. If $k=2$, then $m_{k+1}=m_3=0$, which does not satisfy $1\leqslant m_3\leqslant d_3$. Thus, we obtain $k=1$ and $MTS(\delta_0)=(5,3,1,1,1,1,1,1)$.

The remaining sequence $\delta_1=(5,4,3,3,3,3,2,1)-(5,3,1,1,1,1,1,1)=(0,1,2,2,2,2,1,0)$. We first ignore the two zero elements in δ_0 and compute $MTS(\delta_1)$ for the nonzero subsequence $(1,2,2,2,2,1)$. $MTS(\delta_1)$ contains 6 elements. Thus, $m_{k+1}=(2\times 6-2)-\sum_{i=1}^{k}d_i-\sum_{i=k+2}^{N=6}1=5+k-\sum_{i=1}^{k}d_i$. Since $m_{k+1}=m_6=1$ when $k=5$, we obtain $k=5$ Thus, the MTS-seqeunce of the subsequence $(1,2,2,2,2,1)$ is itself $(1,2,2,2,2,1)$. Considering the original sequence δ_1, we obtain $MTS(\delta_1)=(0,1,2,2,2,2,1,0)$. The degree sequence $d=(5,4,3,3,3,3,2,1)$ is decomposed as $d=MTS(\delta_0)+MTS(\delta_1)$.

The two MTS-tree of δ_0 and δ_1 are presented in figure 7. The generation of a simple graph by LSFGR can be viewed as a recursive procedure that constructs MTS-trees. We observe that the graph G generated by LSFGR for the degree sequence $d=(5,4,3,3,3,3,2,1)$ can be decomposed as the union of two MTS-trees T_0 and T_1, namely $G=T_1+T_2$. Given any graphical sequence d, a corresponding subsequence $MTS(d)$ exists. Thus, the occurrence of self-loops or multiple edges in a graph generated by LSFGR is attributed to the failure of the sequence δ_i to remain graphical at a certain recursive step. Now, we can claim that the degree sequence d can be decomposed as the union of MTS-sequences if LSFGR generates a simple graph for d. Otherwise, the graph is not simple.

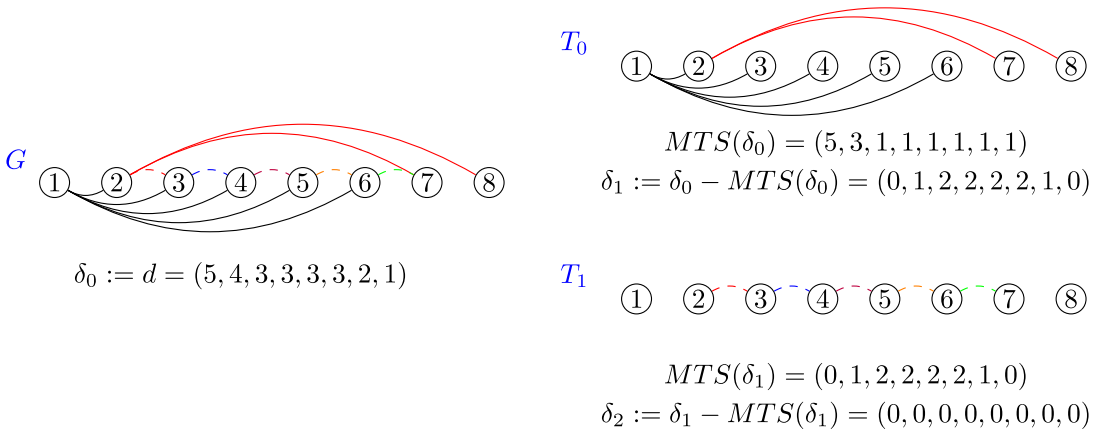


Figure 7. The generation and decomposition of a simple graph $G = T_1 + T_2$ for the degree sequence $d = (5,4,3,3,3,3,2,1)$ by LSFGR.

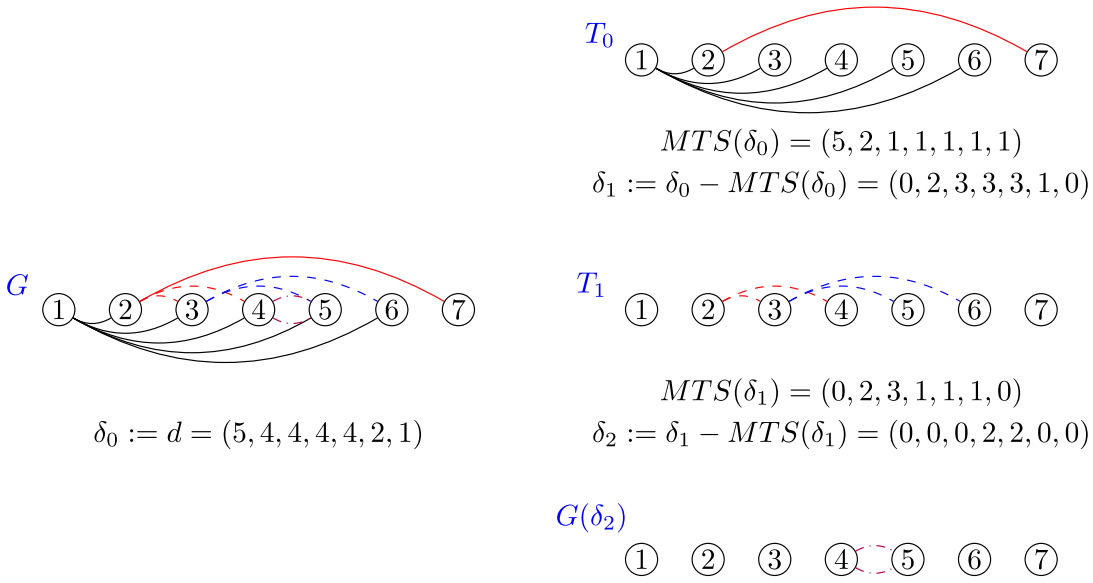


Figure 8. The generation and decomposition of a graph $G = T_1 + T_2 + G(\delta_2)$ with multiple links for the degree sequence $d = (5,4,4,4,4,2,1)$ by LSFGR.

Theorem 3. *Given that the graph G generated by LSFGR of a degree sequence $d = (d_1, d_2, \cdots, d_N)$ is simple, each $\delta_{i+1} := \delta_i - MTS(\delta_i)$ is graphical where $\delta_0 := d$.*

Proof. The fact that LSFGR generates a simple graph $G(\delta_k)$ indicates that the sequence δ_k is graphical. As a result, the sequence $MTS(\delta_k)$ and the tree $T_{MTS(\delta_k)}$ exist. Thus, the subgraph $G(\delta_k) - T_{MTS(\delta_k)}$ generated by LSFGR is simple and realizes the sequence $\delta_{k+1} := \delta_k - MTS(\delta_k)$. Therefore, the sequence δ_{k+1} is graphical. Combining the initial hypothesis that the graph G generated by LSFGR is simple, we prove that every δ_i is graphical. □

We call a degree sequence d *MTS*-decomposable if LSFGR returns a simple graph for the degree sequence d, namely $d = \sum_{i=0} MTS(\delta_i)$. Theorem 3 tells that a realization from LSFGR is not simple if the given degree sequence is not *MTS*-decomposable.

An example with multiple links is illustrated in figure 8. The degree sequence $d = (5,4,4,4,4,2,1)$ is not *MTS*-decomposable because $\delta_2 = (0,0,0,2,2,0,0)$ is not graphical. Two links are added between nodes 4 and 5, leading to that the graph G is not simple.

We point out that the existence of a *MTS* defined in equation (5) is a necessary condition for a degree sequence to be graphical. The converse does not hold: a non-graphical degree sequence may still admit a *MTS*. For the example with self-loops in figure 9, the sequence $d = (4,4,4,4,2,1,1)$ is not *MTS*-

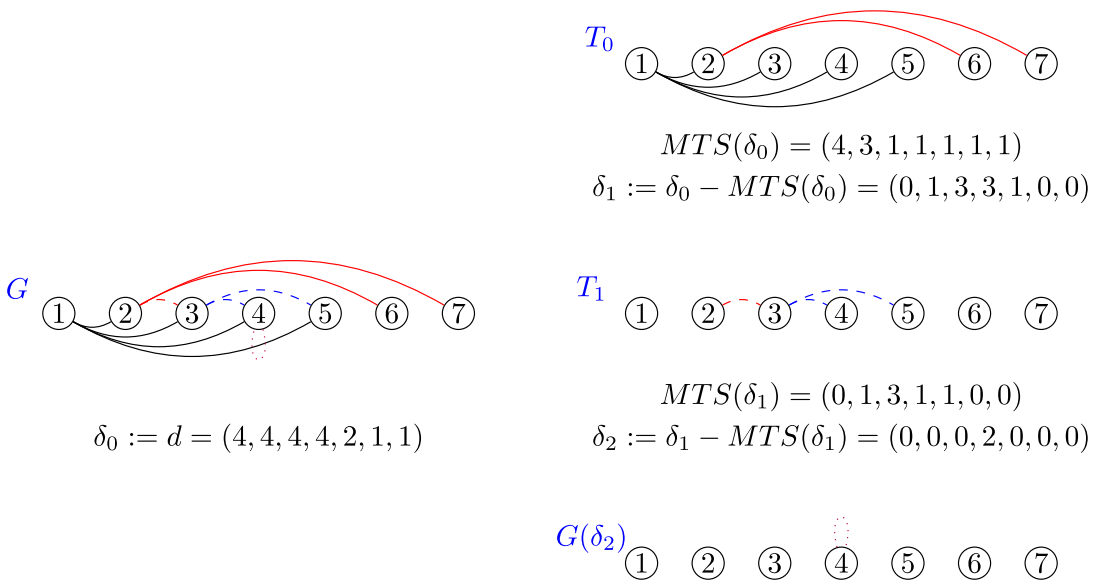


Figure 9. The generation and decomposition of a graph $G = T_1 + T_2 + G(\delta_2)$ with a self-loop for the degree sequence $d = (4,4,4,4,2,1,1)$ by LSFGR.

decomposable because the sequence $\delta_1 = (0,1,3,3,1,0,0)$ is not graphical. Nevertheless, computing a *MTS* for δ_1 is still possible, specifically $MTS(\delta_1) = (0,1,3,1,1,0,0)$.

In any realization generated by LSFGR, at most one node has self-loops, provided that the degree sequence satisfies that the sum of its elements is even.

Property. Given a degree sequence $d = (d_1, d_2, \cdots, d_N)$, the graph generated by LSFGR has at most one node with self-loops.

Proof. We assume that the graph G generated by LSFGR has two nodes i and j with self-loops and $i < j$ without loss of generality. In one step k, the node i is the hub and the self-loop $i \sim i$ is added. A self-loop $j \sim j$ is added for node j in another step k^*. We have $k < k^*$ because $i < j$. The remaining degree of node j in step k should be larger than 1, otherwise, node j cannot have a self-loop in a later step k^*, indicating that node i should be connected to node j, not itself, in the step k, which contradicts our assumption. □

Next, we address the problem of modifying LSFGR to generate simple graphs for non-*MTS*-decomposable degree sequences. A straightforward approach might involve applying suitable link rewiring methods. However, we propose an alternative method that integrates the Havel-Hakimi algorithm with LSFGR. This approach leverages the known locations of self-loops and multiple edges, which typically arise when a certain sequence δ_i is not graphical.

Given a non-*MTS*-decomposable degree sequence d, we decompose d as

$$d = \sum_{i=0}^{k-1} MTS(\delta_i) + \delta_k \tag{6}$$

where δ_k is the last remaining sequence that remains graphical.

The graph generation process is then divided into two stages. LSFGR is applied to the subsequence $\sum_{i=0}^{k-1} MTS(\delta_i)$, while the Havel-Hakimi algorithm is used to generate a simple subgraph corresponding to the remaining sequence δ_k. The final graph G corresponding to d is constructed as the union of the two subgraphs produced by these two respective methods. The complete framework is illustrated in the flowchart in figure 10, where HH represents a Haver-Hakimi algorithm. If a given sequence is *MTS*-decomposable, then LSFGR is applied directly. Otherwise, LSFGR is used to construct a subgraph of degree sequence $\sum_{i=0}^{k-1} MTS(\delta_i)$, followed by application of the Havel-Hakimi algorithm to the remaining graphical subsequence δ_k.

4.2.2. Simulation

In this section, we identify specific families of degree sequences for which LSFGR produces either simple or non-simple realizations. Furthermore, we present simulations characterizing the occurrence of self-loops and multiple edges generated by LSFGR.

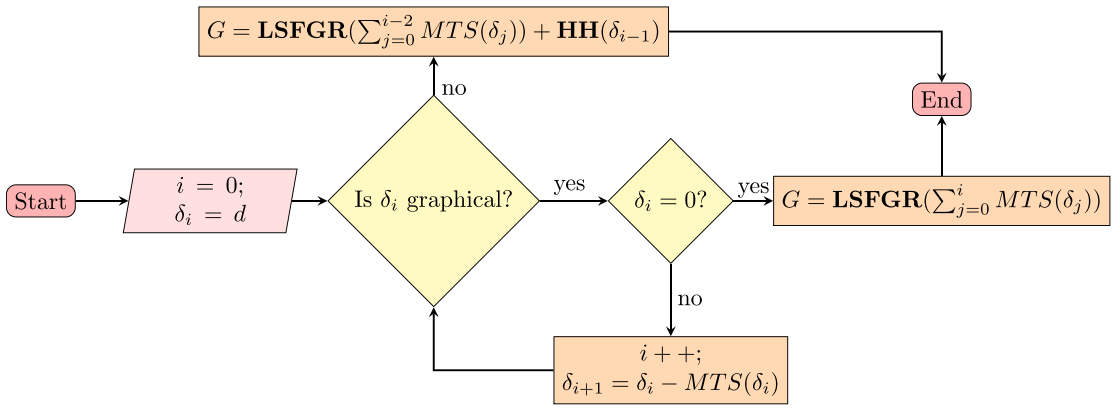


Figure 10. The framework of the graph realization procedure for graphical degree sequences.

Property. A graph generated by LSFGR is simple and connected if the given degree sequence $d = (d_1, d_2, \cdots, d_N)$ satisfies $\sum_{i=1}^{N} d_i = 2N-2$ or $\sum_{i=1}^{N} d_i = 2N$.

Proof. If $\sum_{i=1}^{N} d_i = 2N-2$, then the sequence d corresponds to the degree sequence of a tree with N nodes, as the sum of degrees in any tree is $2(N-1)$. Since the LSFGR algorithm guarantees the generation of connected realizations for potentially connected degree sequences, the resulting graph must be a tree. If $\sum_{i=1}^{N} d_i = 2N$, then LSFGR initially constructs a spanning tree with a degree sequence defined in equation (5), which ensures connectivity, as shown in figure 6. Subsequently, the algorithm adds a link between nodes K+ 1 and $K+2$ to reach the degree sequence d. As the link $(K+1) \sim (K+2)$ is added between two distinct nodes and does not introduce multiple edges or self-loops, the final graph remains simple and connected. □

Property. A graph generated by LSFGR always has multiple links or self-loops if the given degree sequence d satisfies

$$d = \left(d_1 = k+m,\ d_2 = k+m,\ \underbrace{d_3 = \cdots = d_{k+1} = k}_{k-1},\ \underbrace{d_{k+2} = \cdots = d_N = 2}_{2m} \right),$$

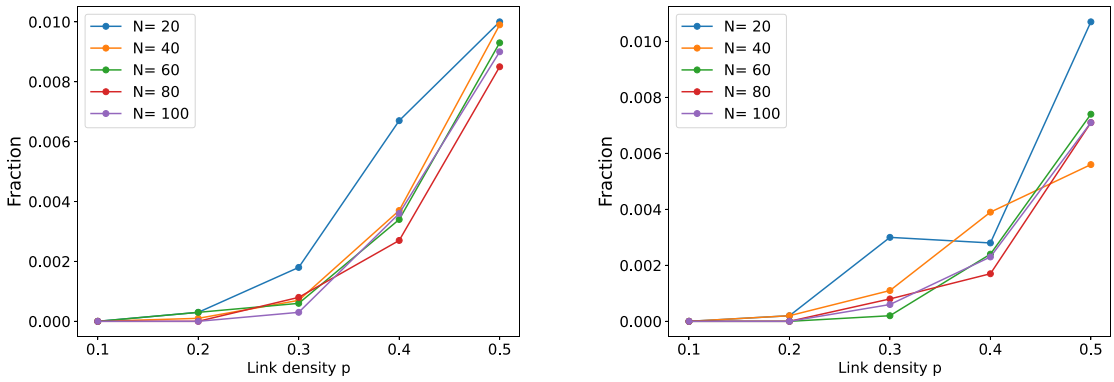
where $0 < 2m < (k-2)(k-1)$.

Proof. First, we show that the above family of degree sequences is graphical. The simple realization is constructed by first creating the clique K_{k+1} using the nodes of degree k and $k+m$, whose degree sequence is $(\underbrace{k, \cdots, k}_{k+1})$. The remaining degree sequence is $t' = (m, m, \underbrace{0, \ldots, 0}_{k-1}, \underbrace{2, \ldots, 2}_{2\text{ m}})$. Then we connect the two nodes of final degree $k+m$ each to m different nodes of remaining degree 2. The remaining degree sequence is $t'' = (0, 0, 0, \ldots, 0, \underbrace{1, \ldots, 1}_{2\text{ m}})$. Then, we separate the 2 m nodes of remaining degree 1 into m disjoint pairs and connect those. The result is a connected simple graph, which means that every degree sequence in the family is graphical. Because of this construction, we will call the first $k+1$ nodes the 'clique nodes'.

Next, we show that LSFGR will always fail to produce simple graphs for this degree sequence family. First, the first node is connected to the k other clique nodes and then to the first m nodes of final degree 2. Then, the second node is connected to the other m nodes of final degree 2 and the $k-1$ clique nodes of final degree k. Now, the intermediate degree sequence is $d^{'} = (k+m,\ k+m,\ \underbrace{2, \cdots, 2}_{k-1},\ \underbrace{1, \cdots, 1}_{2m})$ and the remaining degree sequence is $t = d - d^{'} = (0,\ 0,\ \underbrace{k-2, \cdots, k-2}_{k-1},\ \underbrace{1, \cdots, 1}_{2m})$. Since $2\text{ m} < (k-2)(k-1)$, not all nodes of remaining degree $k-2$ can connect only to nodes of remaining degree 1. Therefore, we will run out of nodes of remaining degree 1 to connect to before the degree sequence is realized.

When the nodes of the remaining degree 1 run out, there are two cases: (I) $2m < k-2$; (II) $2m \geqslant k-2$.

(I) The remaining degree sequence is $t = (0, 0, k-2-2m, \underbrace{k-2, \cdots, k-2}_{k-2}, \underbrace{0, \cdots, 0}_{2m})$, where $n = k-1$ nodes are left. Since $k-2$ nodes have remaining degree $k-2$ among $k-1$ left nodes, the only simple realization of the remaining degree t is a clique K_{k-1} in which all nodes have degree $k-2$. However, $t_3 = k-2-2m < k-2$ implies there must be multiple links or self-loops among the $k-2$ nodes of remaining degree $k-2$.



(a) multiple links (b) self loops

Figure 11. The fraction of realizations with (a) multiple links and (b) self-loops generated by LSFGR out of 10^4 realizations for potentially connected degree sequences from graphs $G(N, L)$ versus the graph size N and the link density $p = \frac{L}{L_{\max}}$.

(II) The remaining degree sequence is $t = (0, \cdots, 0, \underbrace{t_i, k-2, \cdots, k-2}_{n \leqslant k-2}, \underbrace{0, \cdots, 0}_{2m})$, where $n \leqslant k-2$ nodes are left. There is no simple graph that contains $n \leqslant k-2$ nodes and a node with degree $k-2$. Combining the two cases indicates that the graphs generated from LSFGR for the defined degree sequences are not simple. □

We estimate the prevalence of self-loops and multiple links produced by LSFGR by simulations on 10^4 potentially connected degree sequences that correspond to 10^4 random graphs $G(N, L)$. We consider graph sizes $N \in \{20, 40, 60, 80, 100\}$ and vary the number of links $L = L_{\max} p$, where $L_{\max} = \frac{N(N-1)}{2}$ represents the number of links in a complete graph with N nodes and $p \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$ denotes the link density. Figure 11 presents the fraction of graphs generated by LSFGR with at least one self-loop or multiple links. The number of graphs that contain self-loops or multiple links increases as link density p grows. Similarly, when the graph size $N \geqslant 40$, the number of graphs containing self-loops also exhibits minimal variation with respect to N. For a fixed link density p, the proportion of graphs with multiple links remains fairly constant across different graph sizes N. The occurrence of both self-loops and multiple edges rises more sharply as link density p increases.

4.3. LSFGR assortativity

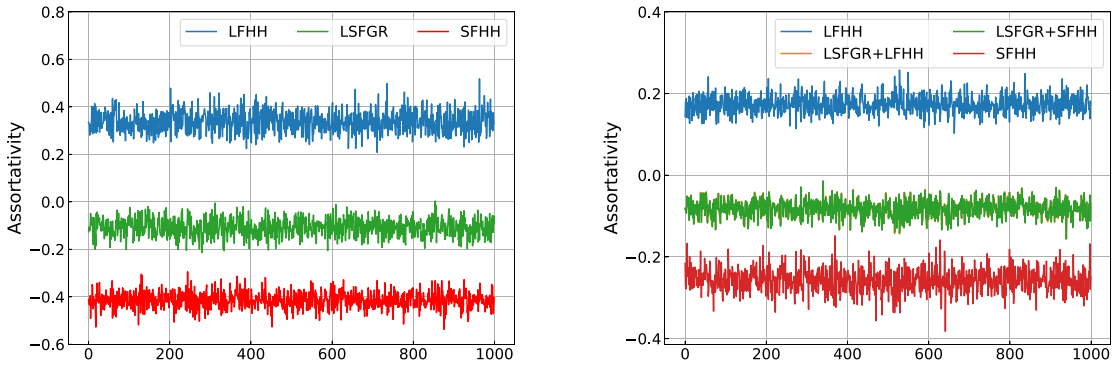
As discussed in section 2.2, LFHH genrates graphs with high assortativity by preferentially connecting high-degree nodes to other high-degree nodes. In contrast, SFHH tends to generate graphs with low assortativity by favoring connections between low-degree and high-degree nodes. In the case of LSFGR, the hub node has the highest final degree among the nodes with remaining degree. During the graph construction process, each hub node iteratively connects to the node with the smallest intermediate degree, which effectively results in each hub connecting to nodes from a high degree to a low degree until the hub reaches its final degree. Consequently, the graphs generated by LSFGR exhibit moderate assortativity.

Figure 12 depicts the assortativity of 10^3 graphs of $N = 50$ nodes, which are generated by three algorithms: LFHH and SFHH and LSFGR. The underlying given degree sequences are derived from 10^3 connected random graph $G(N, L)$, where $L = L_{\max} p = \frac{N(N-1)}{2} p$ and p denotes the link density.

We consider both *MTS*-decomposable and non-*MTS*-decomposable degree sequences in our experiments. For each *MTS*-decomposable degree sequence, three algorithms are directly applied to construct three simple graphs. For a non-*MTS*-decomposable degree sequence, the generation process follows the procedure outlined in Flowchart 10. A non-*MTS*-decomposable degree sequence is decomposed as $d = \sum_{i=0}^{k-1} MTS(\delta_i) + \delta_k$ in equation (6). LSFGR is applied to the subsequence $\sum_{i=0}^{k-1} MTS(\delta_i)$, while both LFHH and SFHH algorithms are independently applied to the subsequence δ_k. As a result, four graphs are generated for each non-*MTS*-decomposable degree sequence.

For *MTS*-decomposable degree sequences, we set the link density $p = 0.3 \approx 5p_c$ where $p_c = \frac{\log N}{N}$ denotes the threshold for the emergence of connectedness in $G(N, p)$. As illustrated in figure 12(a), graphs generated by LFHH and SFHH exhibit approximately opposite assortativity values, around 0.4 and -0.4, respectively. In contrast, graphs generated by LSFGR exhibit neutral assortativity values, approximately -0.1, which is consistent with our theoretical expectation that LSFGR produces graphs with moderate assortativity.

For non-*MTS*-decomposable degree sequences, we choose a higher link density $p = 0.6$, as non-*MTS*-decomposable degree sequences are more likely to occur for graphs with higher degree density. In



(a) MTS-decomposable sequences (b) non-MTS-decomposable sequences

Figure 12. The assortativity of 10^3 realizations $G(N, L)$ of (a) *MTS*-decomposable sequences from three algorithms (LFHH, LSFGR and SFHH) with $N = 50$, $L = L_{\max}p$ and $p = 0.3$; (b) non-*MTS*-decomposable sequences from four algorithms (LFHH, SFHH, LSFGR+LFHH and LSFGR+SFHH) with $N = 50$, $L = L_{\max}p$ and $p = 0.6$; At this scale, the orange line (LSFGR+LFHH) is nearly overlapped by the green line (LSFGR+SFHH) and not clearly visible in the figure.

this case as well, the LSFGR algorithm consistently generates graphs with moderate assortativity, regardless of whether LFHH or SFHH is applied to the subsequence δ_k, which is demonstrated in figure 12(b). Furthermore, the close overlap of the corresponding yellow and green lines in figure 12(b) suggests that the choice between LFHH and SFHH, when applied to the remaining degree sequence δ_k, has minimal impact on the assortativity in the configuration for non-*MTS* decomposable degree sequences. The assortativity of realizations corresponding to non-*MTS* decomposable degree sequences appears to be determined primarily by the subgraph produced by the LSFGR.

We also analyze the modularity and the clustering coefficient of the graphs generated from LFHH, SFHH and LSFGR. The modularity [1] quantifies the strength of community structure within a graph, which is defined as

$$m = \frac{1}{2L}\sum_{i=1}^{N}\sum_{j=1}^{N}(a_{ij} - p_{i,j})\sum_{k=1}^{C}\mathbf{1}_{\{i,j\in C_k\}},$$

where C is the number of communities and C_k denotes the community k. The indicator function $1_{\{i,j\in C_k\}}$ means that only nodes in the same community C_k contribute to modularity. The term $p_{i,j}$ represents the probability that a link would exist between nodes i and j at random but respecting node degrees [24]. The author in [25] have proposed a accurate probability $p_{i,j}$ as

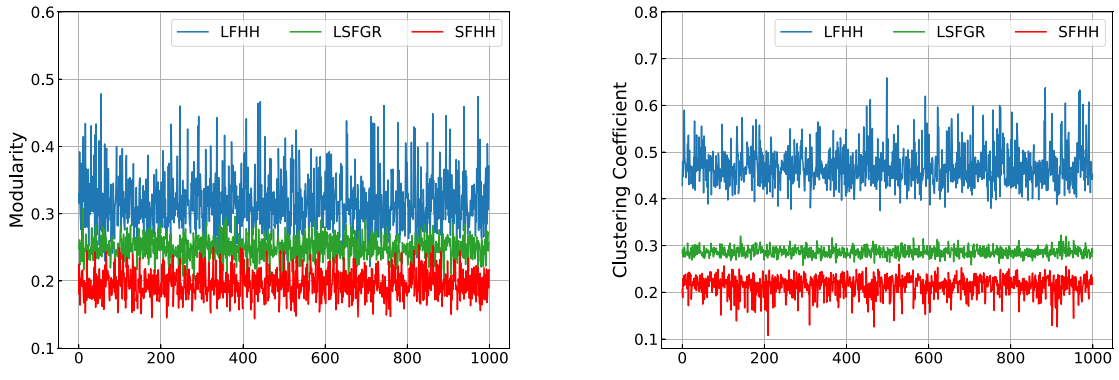
$$p_{i,j} = \begin{cases} \frac{d_i d_j\left(L^c - d_i^c - d_j^c + 1\right)}{d_i d_j\left(L^c - d_i^c - d_j^c + 1\right) + d_i^c d_j^c\left(L - d_i - d_j + 1\right)}, & i \neq j \\ 0, & i = j \end{cases}$$

where $d_i^c = (N-1) - d_i$ and $L^c = L_{\max} - L$ and the superscript 'c' refers to the complement of the graph [2].

The clustering coefficient [26] measures the tendency of nodes to form groups, reflecting the likelihood that a node's neighbors are also connected to each other, calculated as

$$C_G = \frac{6 \times \text{the number of triangles}}{\text{number of connected triples}}.$$

In our study, figure 13 presents the modularity and the clustering coefficient for 10^3 graphs generated by three algorithms: LFHH, SFHH and LSFGR. The underlying degree sequences are *MTS*-decomposable degree sequences, derived from random and connected graph $G(N, L)$, where $N = 50$ and $L = L_{\max}p = \frac{N(N-1)}{2}p$ with link density $p = 0.3$. Higher modularity values indicate a more distinct community structure. Using Newman's spectral algorithm [27], figure 13(a) shows that LFHH consistently achieves the highest modularity, indicating superior detection of communities. LSFGR scores moderately, while SFHH has the lowest modularity, suggesting weaker community structure. Figure 13(b) demonstrates that LFHH also attains the highest clustering coefficients, implying better preservation of local connectivity. LSFGR exhibits moderate clustering, and SFHH the lowest, indicating sparser local connections among neighbors.



(a) Modularity (b) Clustering Coefficient

Figure 13. The (a) modularity and (b) clustering coefficient of 10^3 realizations $G(N, L)$ generated by three realization algorithms: LFHH, LSFGR, and SFHH, where $N = 50$, $L = L_{\max}p$, and the link density $p = 0.3$.

5. Conclusion

We address the problem of constructing graphs from degree sequences under structural constraints of connectedness and assortativity. We propose LSFGR, a new realization method that ensures connected graphs for all potentially connected sequences. When a simple realization is not possible, the algorithm allows at most one node with self-loops. LSFGR produces graphs with moderate degree correlations. Our results show that LSFGR effectively supports constrained graph realization.

Data availability statement

No new data were created or analysed in this study.

Acknowledgments

We thank Brian L Chang for his contribution of the simulation of the modularity in this article. Furthermore, we are grateful to Robin Persoons for his useful comments and suggestions regarding this work. Yingyue Ke has been funded by the China Scholarship Council (Grant No. 202206270016). Piet Van Mieghem has been funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (Grant Agreement No. 101019718).

ORCID iD

Yingyue Ke 0009-0006-6801-868X

References

[1] Newman M 2018 *Networks* (Oxford University Press)
[2] Van Mieghem P 2023 *Graph Spectra for Complex Networks* (Cambridge University Press)
[3] Qiu Z, Jokić I, Tang S, Noldus R and Van Mieghem P 2023 Inverse all shortest path problem *IEEE Trans. Netw. Sci. Eng.* **11** 2703–14
[4] Arnold N, Mondragón R and Clegg R 2022 Reconstructing degree distribution and triangle counts from edge-sampled graphs *Int. Conf. Complex Networks And Their Applications* pp 297–309
[5] Horvát S and Modes C 2021 Connectedness matters: construction and exact random sampling of connected networks *J. Phys.: Complex.* **2** 015008
[6] Erdös P and Gallai T 1960 Gráfok előírt fokszámú pontokkal *Mat. Lapok.* **11** 264–74
[7] Hakimi S 1962 On realizability of a set of integers as degrees of the vertices of a linear graph. I *J. Soc. Indus. Appl. Math.* **10** 496–506
[8] Havel V 1955 A remark on the existence of finite graphs *Cas. Pest. Mat.* **80** 477–80
[9] Wang D and Kleitman D 1973 On the existence of N-connected graphs with prescribed degrees ($n \geqslant 2$) *Networks* **3** 225–39
[10] Bassler K, Del Genio C, Erdös P, Miklós I and Toroczkai Z 2015 Exact sampling of graphs with prescribed degree correlations *New J. Phys.* **17** 083052
[11] Taylor R 2006 Contrained switchings in graphs *Comb. Math.* **VIII** 314–36
[12] Kim H, Toroczkai Z, Erdös P, Miklós I and Székely L 2009 Degree-based graph construction *J. Phys. A: Math. Theor.* **42** 392001
[13] Rao A, Jana R and Bandyopadhyay S 1996 A Markov chain Monte Carlo method for generating random (0, 1)-matrices with given marginals *Indian J. Stat. A* **58** 225–42
[14] Del Genio C, Kim H, Toroczkai Z and Bassler K 2010 Efficient and exact sampling of simple graphs with given arbitrary degree sequence *PLoS One* **5** e10012

[15] Noldus R and Van Mieghem P 2015 Assortativity in complex networks *J. Complex Netw.* **3** 507–42
[16] Zhou D, Stanley H, D'Agostino G and Scala A 2012 Assortativity decreases the robustness of interdependent networks *Phys. Rev. E* **86** 066103
[17] Chang S, Piraveenan M and Prokopenko M 2020 Impact of network assortativity on epidemic and vaccination behaviour *Chaos Solitons Fractals* **140** 110143
[18] Foster D, Foster J, Grassberger P and Paczuski M 2011 Clustering drives assortativity and community structure in ensembles of networks *Phys. Rev. E* **84** 066117
[19] Newman M 2002 Assortative mixing in networks *Phys. Rev. Lett.* **89** 208701
[20] Van Mieghem P, Wang H, Ge X, Tang S and Kuipers F 2010 Influence of assortativity and degree-preserving rewiring on the spectra of networks *Eur. Phys. J. B* **76** 643–52
[21] Newman M and Park J 2003 Why social networks are different from other types of networks *Phys. Rev. E* **68** 036122
[22] Johnson S, Torres J, Marro J and Munoz M 2010 Entropic origin of disassortativity in complex networks *Phys. Rev. Lett.* **104** 108702
[23] Guimera R, Mossa S, Turtschi A and Amaral L 2005 The worldwide air transportation network: anomalous centrality, community structure and cities' global roles *Proc. Natl Acad. Sci.* **102** 7794–9
[24] Clauset A, Newman M and Moore C 2004 Finding community structure in very large networks *Phys. Rev. E* **70** 066111
[25] Chang B and Van Mieghem P 2025 Modularity with a more accurate baseline model *Phys. Rev. E* **111** 044317
[26] Van Mieghem P 2014 *Performance Analysis of Complex Networks and Systems.* (Cambridge University Press)
[27] Newman M 2006 Modularity and community structure in networks *Proc. Natl Acad. Sci.* **103** 8577–82