2022

Graph Realizability and Factor Properties Based on Degree Sequence

Daniel John
University of Windsor

Graph realizability and factor properties based on degree sequence

By

Daniel John

A Thesis
Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science
at the University of Windsor

Windsor, Ontario, Canada

2022

Graph realizability and factor properties based on degree sequence

By

Daniel John

Approved by

M. S. Monfared
Department of Mathematics and Statistics

A. Biniaz
School of Computer Science

A. Mukhopadhyay, Advisor
School of Computer Science

December 19, 2022

DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

A graph is a structure consisting of a set of vertices and edges. Graph construction has been a focus of research for a long time, and generating graphs has proven helpful in complex networks and artificial intelligence.

A significant problem that has been a focus of research is whether a given sequence of integers is graphical. Havel and Hakimi stated necessary and sufficient conditions for a degree sequence to be graphic with different properties. In our work, we have proved the sufficiency of the requirements by generating algorithms and providing constructive proof.

Given a degree sequence, one crucial problem is checking if there is a graph realization with k-factors. For the degree sequence with a realizable k-factor, we analyze an algorithm that produces the realization and its k-factor. We then generate degree sequences having no realizations with connected k-factors. We also state the conditions for a degree sequence to have connected k-factors.

In our work, we have also studied the necessary and sufficient conditions for a sequence of integer pairs to be realized as directed graphs. We have proved the sufficiency of the conditions by providing algorithms as constructive proofs for the directed graphs.

DEDICATION

To all the people who have supported me in my whole academic journey.

ACKNOWLEDGEMENTS

I want to express my gratitude to my supervisor Dr. Asish Mukhopadhyay for his continuous interaction and support in teaching me concepts and helping me to write this thesis. His ideas and theories have been a driving force for me to strive ahead in this area of research. I would also like to thank my thesis committee members, Dr. Ahmad Biniaz and Dr. Mehdi Sangani Monfared, for their valuable time for this thesis.

I also want to express my gratitude toward the Computer Science faculty members and staff who have helped me in my journey of writing this thesis.

Finally, I would like to thank my parents, my sister, my cousin, and their family for helping me in every way possible and encouraging me to move forward.

CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

Due to their versatility, graphs have been studied for a long time. Graphs are used in wide range of fields that includes modelling of optimal network connections, generating protein-protein interactions, building complex networks etc. Due to different fields of applications, several problems related to graph operations such as realizing degree sequences, number of possible realizations, unique graphs, etc. are major topics of study. In this thesis we study different necessary and sufficient conditions used for realizing integer sequences as directed and undirected graphs while also discussing the k-factor properties of different degree sequences.

1.1 Preliminaries

A graph is denoted by $G = (V, E)$ and is a structure consisting of a set of vertices (V) and a set of edges (E) connecting the vertices. Two vertices connected using an edge are known as adjacent vertices. Figure 1.1 shows a graph with 5 vertices and 6 edges. Each vertex has degrees associated to it and it is equal to the number of edges incident to that vertex.

The degree of a vertex v in an undirected graph is represented using d_v. For n vertices

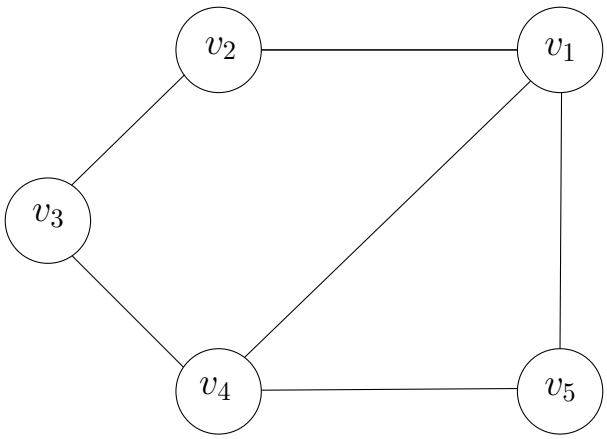

Figure 1.1: *Example of a graph*

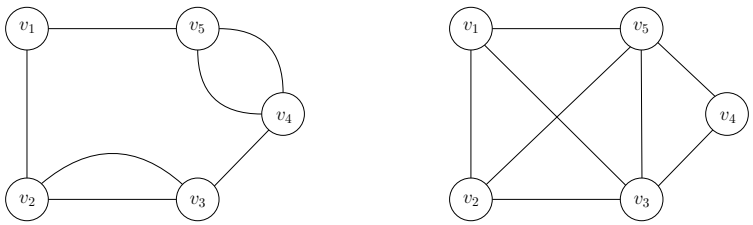

Figure 1.2: *(a) Multigraph (b) Simple graph*

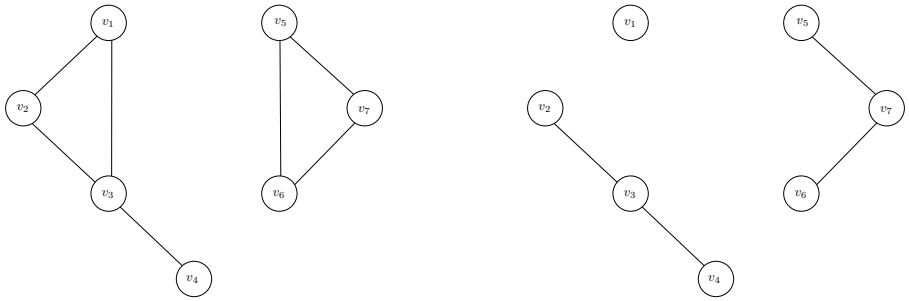

Figure 1.3: *(a) Graph G (b) Subgraph of G*

we represent the sequence of degrees in the form $d_1, d_2, \ldots, d_n$. The vertex degrees can be represented in an increasing or decreasing order as seen in [5] and [7].

For directed graphs the degree of a vertex is represented using pairs of out-degrees and in-degrees (v_{out}, v_{in}). The sequence of the pairs are arranged in lexicographically increasing or decreasing order.

1.2 Graph Terminologies

Some of the terminologies used in this thesis are:

1. **Multigraphs:** are the graphs having multiple edges between at least one pair of vertices. Figure 1.2(a) shows an example of a multigraph, where vertex pairs (v_2, v_3) and (v_4, v_5) have multiple edges between them.

2. **Simple graphs:** are the graphs that do not have self loops or multiple edges. Figure 1.2(b) shows an example of a simple graph.

3. **Subgraph:** A graph $H = (V', E')$ is a subgraph of G if $E' \subseteq E$ and $V' \subseteq V$. The vertices at the ends of each edge in E' should be present in V'. Figure 1.3 shows an example of a graph G and a subgraph of G.

4. **Maximally connected subgraph of G:** is a subgraph $H = (V', E')$ in which no more edges from E can be added to E'.

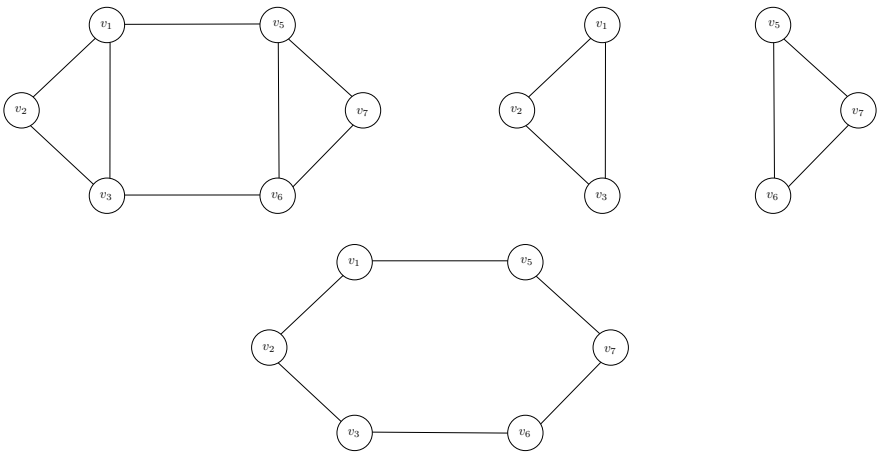

Figure 1.4: *(a) Graph G (b) Cycles in G (c) A Hamiltonian cycle of G*

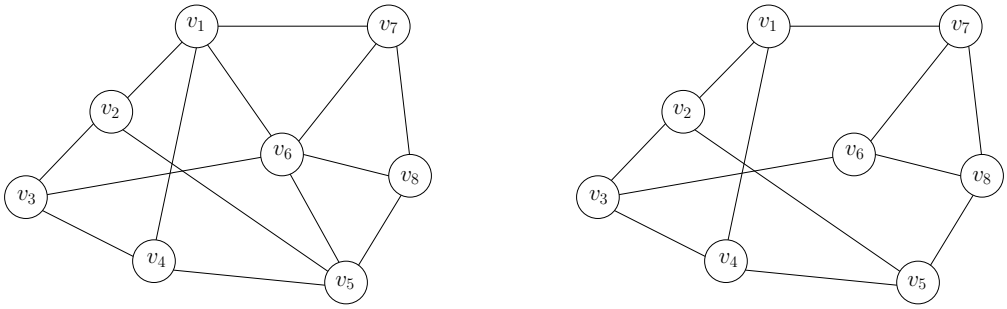

Figure 1.5: *(a) Graph G (b) A 3-factor of G*

5. **Component of a graph G:** is a maximally connected subgraph that have path from each vertex to all other vertices. In figure 1.3(a) the subgraph along the path $v_3v_1v_2v_3v_4$ is a component of G.

6. **A path** is a sequence of edges that connects a set of adjacent vertices. In figure 1.2(b) (v_2, v_5, v_4, v_3) is a path.

7. **A cycle in G:** is a path of edges that begins and ends at the same vertex. Figure 1.4(a) and (b) shows graph G and some of the possible cycles of in G.

8. **A Hamiltonian cycle in G:** is a cycle that visits each vertex of G exactly once. Figure 1.4(c) shows a Hamiltonian cycle of G.

9. **Spanning subgraph of G:** is a subgraph having all the vertices in V and $E' \subseteq E$.

10. **k-factor of G:** is a k-regular spanning subgraph of G, where $1 \leq k \leq min_i(d_i)$. Figure 1.5 shows graph G and a 3-factor of G.

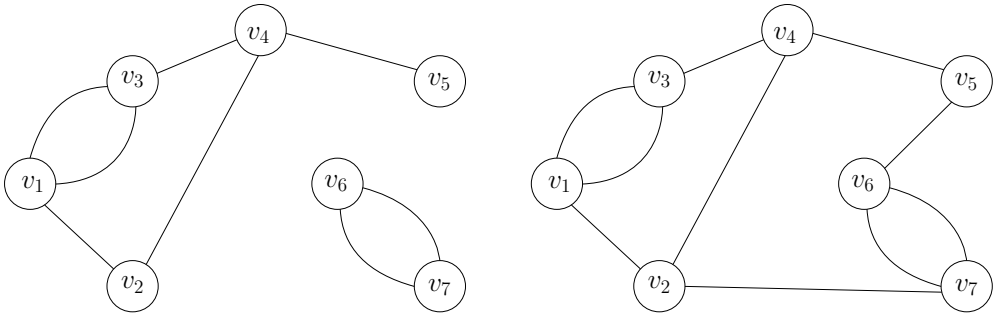

Figure 1.6: *(a) A multigraph without loops (b) A connected multigraph without loops*

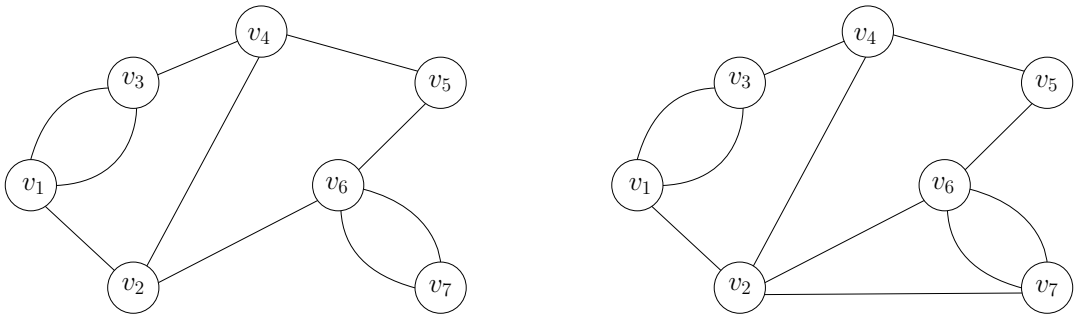

Figure 1.7: *A connected (a) separable multigraph (b) non-separable multigraph*

1.3 Constrained graphical sequences

There are several types of graphs that have different characteristics. Degree sequences with constraints can be used to realize different types of graphs. In the chapter 2 of this thesis, we focus on four types of constrained graphical sequences and their realizations.

1. **Multigraphs without loops:** These graphs can have multiple edges between a pair of vertices. However, there will be no self loops for any vertex, where a loop is an edge from a vertex to itself. Figure 1.6(a) is a multigraph without loops.

2. **Connected multigraphs:** These graphs have multiple edges and have a single component. Figure 1.6(b) is a connected multigraph.

3. **Connected separable multigraphs:** These are connected multigraphs that has one cut vertex. The cut-vertex when removed will make the graph non-connected. Figure 1.4(a) shows an example of connected separable multigraph.

4. **Connected non separable multigraph:** These are connected multigraphs that do not have a cut vertex which can be removed for non-connected multigraph. Figure 1.4(b) is an example of a connected non-separable multigraph.

1.4 Operations on graphs

For a degree sequence to be realizable as a graph of certain properties, some operations needs to be performed while generating the graph such that the degrees of the vertices

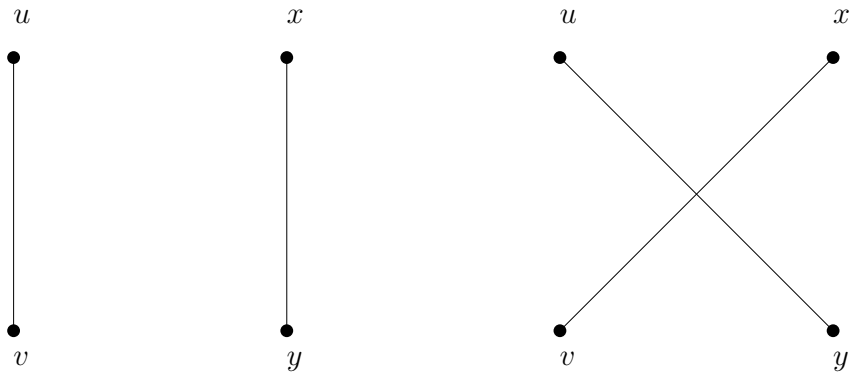

Figure 1.8: *Edge switching for two components*

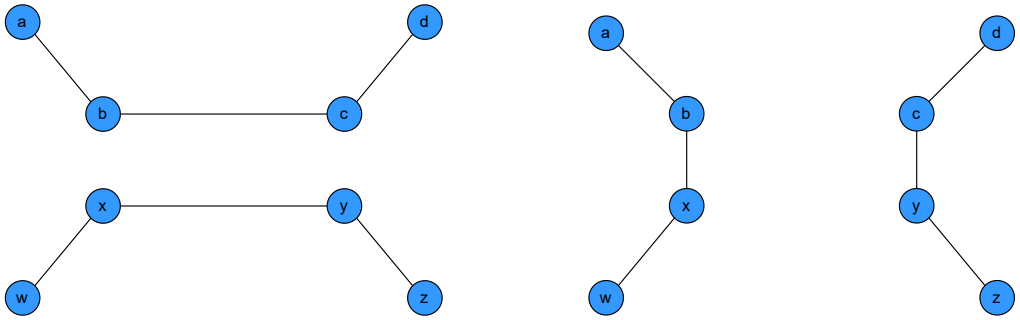

Figure 1.9: *Edge switching for two components without cycles*

are unchanged when rewiring is done. These operations helps us to realize graphs easily. One of the well defined operation for rewiring the edges in a graph is edge switching.

1.4.1 Edge switching

Let $\{u, v\}$ and $\{x, y\}$ be two independent edges between vertices u, v and x, y respectively in graph G. These edges can be replaced with edges $\{u, y\}$ and $\{v, x\}$ (as shown in figure 1.8) while preserving the degrees of vertices.

Switching edges that are not part of a cycle

If edges $\{u, v\}$ and $\{x, y\}$ belong to different components of the graph and do not belong to any cycle in graph G, after switching the edges to $\{u, y\}$ and $\{v, x\}$, the resulting graph will be having two different components. Figure 1.9 shows the result of edge switching in case of two edges in different components and not a part of any cycles.

Switching edges that are part of a cycle

If edges $\{u, v\}$ and $\{x, y\}$ belong to different components of the graph and if any one of the edge belong to any cycle in graph G, after switching the edges to $\{u, y\}$ and $\{v, x\}$, the resulting graph will have a merged component as shown in figure 1.10.

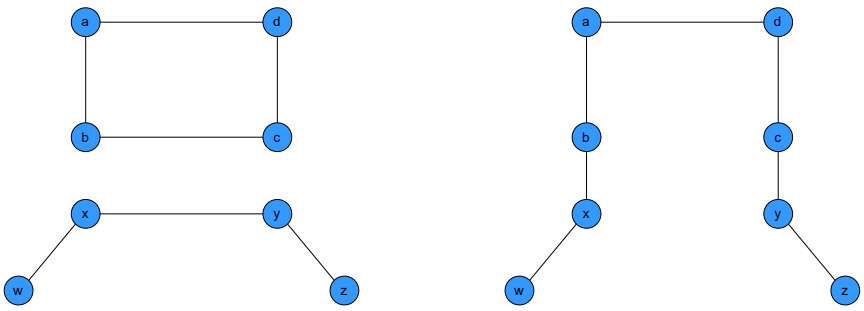

Figure 1.10: *Edge switching for two components, with one edge as part of a cycle*

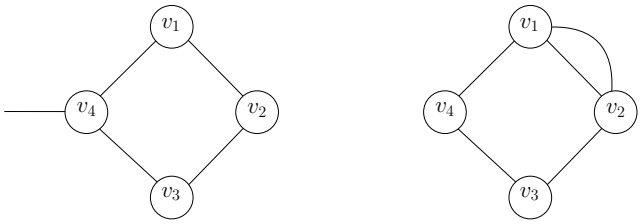

Figure 1.11: *(a) Unrealizable sequence* $(3, 2, 2, 2)$ *(b) Realizable sequence* $(3, 3, 2, 2)$

1.5 Graphicality of a sequence

A sequence of non-negative integers is graphic if at least one graph exists with vertex degrees corresponding to the degree sequence. In the figure 1.11, the realization of the sequence $(3, 2, 2, 2)$ is not graphic, whereas the sequence $(3, 3, 2, 2)$ is graphic as it can be realized as a graph.

Erdős-Gallai [5] stated that a list of non-negative integers $\{d_1, d_2, \ldots, d_n\}$ is graphic if and only if the sum of the integers is even and for each integer $1 \leq k \leq n-1$

$$\sum_{i=1}^{k} d_i \leq k(k-1) + \sum_{i=k+1}^{n} min(k, d_i) \tag{1.1}$$

The equation 1.1 divides the set of integers into two parts with one part as the set of integers with indices less than k and the second part as the set of integers with indices greater than k. This equation specifies the left side of the inequality is the sum of first k terms and it should be less than or equal to $k(k-1) + \sum_{i=k+1}^{n} min(k, d_i)$. Here $k(k-1)$ denotes the sum of all the degrees when all the vertices with the indices less than the k, when the vertices form a complete graph. $\sum_{i=k+1}^{n} min(k, d_i)$ denotes the upper limit of the sum of all the outgoing edges from the subgraph formed by the first set of integers.

The necessity of the Erdős-Gallai Inequality(EGI) is straightforward. Tripathi et. al [13] proved the sufficiency of EGI constructively by designing an algorithm that realizes graph with the given degree sequence by using this inequality.

1.5.1 Problem statement

Erdős-Gallai Inequality states the conditions for generating simple graphs, and Tripathi et al. gave a constructive proof of graph generation for the simple graphs based on EGI. However to generate graphs of other classes Hakimi et al. [7] stated necessary and sufficiency conditions. In this thesis we take the cue from Tripathi's constructive proof to generate algorithms and constructively proving the conditions for each cases of graphs specified by Hakimi.

1.5.2 Prior works

Graph realizations using degree sequences have a variety of applications. Several problems related to it have been studied in the past.

Koren [9] studied the sequences with unique graphs for the class of simple graphs. Bender et al. [2] determined the asymptotic bound on the number of graphs realizable using the degree sequences. Meanwhile, Arman et al. [1] studied the problem of obtaining a graph realization uniformly at random for a given sequence.

1.6 Thesis organization

This thesis is divided into four chapters:

Chapter 1: In the first chapter, we define graphs, some associated terminologies used throughout and some operations that can be performed to change the structure of graphs while preserving the vertex degrees. We also discuss some previous works and the motivation for the thesis.

Chapter 2: In the second chapter, we take a cue from Tripathi's constructive proof of the sufficiency of the EGI and provide proof for by generating algorithms to construct graphs for the following types of the graph given a sequence of non-negative integers:

1. A multigraph realization
2. A connected multigraph realization
3. A connected non-separable multigraph
4. A connected separable multigraph

Chapter 3: In the third chapter, we will analyze an elegant algorithm to find a graphic realization and k-factor for a given degree sequence. We will also generate a graphical sequence without any realization for connected k-factors and conditions to generate graphical sequences with connected k factors.

Chapter 4: In the fourth chapter, we look into the conditions by Fulkerson [6] for directed graphs and prove the sufficiency of those conditions constructively using different cases and by generating an algorithm to realize graphs using the degree pair sequences.

Chapter 2

Realizations of undirected graphs

2.1 Introduction

In this section, we look at some theorems by Hakimi [7] that give necessary and sufficient conditions for the graphic realization of a sequence of non-negative integers. In what follows, by a generation algorithm we will mean an algorithm that generates a graph that realizes the given sequence. For each of these theorems, we propose a generation algorithm whose existence and correctness is guaranteed by the sufficient conditions.

2.2 Generating a loopless multigraph

Let $d =< d_1, d_2, \ldots, d_n >$ be a sequence of positive integers such that $d_1 \leq d_2 \leq d_3 \leq \cdots \leq d_n$.

Theorem 1 *[7] The necessary and sufficient conditions for positive integers $d_1, d_2, \ldots, d_n$ to be realizable (as the degrees of the vertices of a linear graph) are:*
(i)$\Sigma_{i=1}^{n} d_i = 2e$*, where e is an integer*
(ii)$\Sigma_{i=1}^{n-1} d_i \geq d_n$

A graph whose degrees satisfy the above conditions can have parallel edges, need not be connected or have a vertex separator (cut-vertex).

2.2.1 Algorithm 1

Our algorithm is recursive. First, we consider the base cases. When $n = 2$, we must have $d_1 = d_2$ from the assumption that $d_1 \leq d_2$ and the second condition in the Theorem 1 above. A graph with two vertices and d_1 edges between the two vertices is a realization.

When $n = 3$, it is easy to see that the following equations given the numbers of edges, n_{ij}, between vertices i and j, $1 \leq i < j \leq 3$.

$$
\begin{aligned}
n_{12} &= \frac{(d_1 + d_2 + d_3) - 2d_3}{2} \\
n_{13} &= \frac{(d_1 + d_2 + d_3) - 2d_2}{2} \\
n_{23} &= \frac{(d_1 + d_2 + d_3) - 2d_1}{2}
\end{aligned} \tag{2.1}
$$

It is then easy to construct the graph. When $n > 3$, we reduce our construction to the base cases this way. Consider the derived sequence $d_2, d_3, \ldots, d_n - d_1$ of length $n-1$. If $d_n - d_1 \neq 0$, it is easily seen that either $d_n - d_1$ or d_{n-1} is the largest of the reduced sequence and satisfies all the conditions of Theorem 1. We now recursively construct a graph for the reduced sequence, add a vertex 1 to the graph and join it to the $n-1$-th vertex with d_1 parallel edges to obtain a graph, satisfying the given degree sequence.

If $d_n - d_1 = 0$, we recursively construct a graph on $n-2$ vertices for the sequence $d_2, d_3, \ldots, d_{n-1}$, add two new vertices to this graph and join them with d_1 parallel edges.

Algorithm 1 shows the steps of generating graph for a given Integer sequence. Here are the outputs for the sequences $d = (2, 2, 2, 2, 3, 3)$ and $d = (1, 1, 1, 3, 4, 4, 4, 4, 4)$

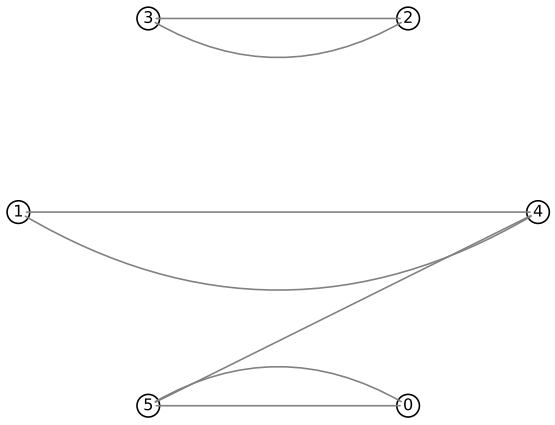

Figure 2.1: *A graph realizing the sequence d = (2, 2, 2, 2, 3, 3)*

2.2.2 Correctness

For $n = 3$, with the help of Equation 2.1, we obtain the numbers of edges n_{ij} between vertices i and j to obtain a graph that realizes the degree sequence $< d_1, d_2, d_3 >$. For $n \geq 3$, assume that we can obtain a correct realization for smaller values of $n \geq 3$.

Algorithm 1 *looplessMultigraph*(v_i)

Input: Non-negative, non-increasing integer sequence
Output: Looples multigraph
if $n = 2$ **then**
  for $i = 1 \rightarrow v_1[0]$ **do**
    Add edge (v_1, v_2)
  end for
  return
end if
if $n = 3$ **then**
  Compute
  $n_{12} = \frac{(d_1+d_2+d_3)-2d_3}{2}$
  $n_{13} = \frac{(d_1+d_2+d_3)-2d_2}{2}$
  $n_{23} = \frac{(d_1+d_2+d_3)-2d_1}{2}$
  Add n_{12} edges between v_1, v_2
  Add n_{23} edges between v_2, v_3
  Add n_{34} edges between v_3, v_4
  return
end if
$v_n[0] = v_n[0] - v_1[0]$
$v_{(1,n)} = v_1[0]$
Shift v_n
remove v_1 from list
looplessMultigraph(v_i)
for $i = 0 \rightarrow v_{(1,n)}$ **do**
  Add edge (v_1, v_n)
end for

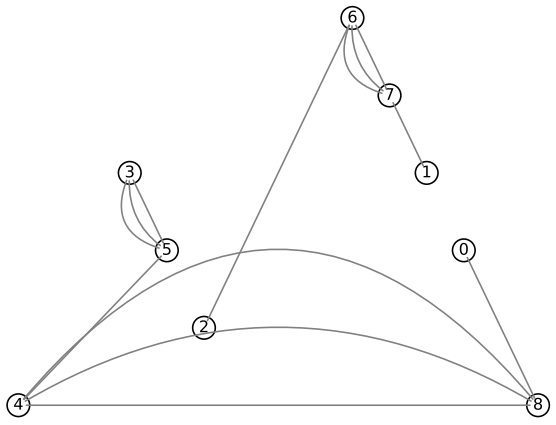

Figure 2.2: *A graph realizing the sequence d = (1, 1, 1, 3, 4, 4, 4, 4, 4)*

The proposed algorithm reduces the construction to a degree sequence of length $n-1$ or $n-2$. In each case, by the assumption above the construction can be carried out correctly, and then the resulting graph augmented by introducing additional $d_n - d_1$ edges between one new vertex, representing 1, and the vertex n or d_1 edges between two new vertices that represent the vertices 1 and n.

The generated graph is loop-free as no loops are created during the base case, and none are created when we return from a recursive call, corresponding to the two cases discussed above.

2.2.3 Complexity

The complexity of the algorithm is proportional to the size of the graph that is generated. This is bounded above by $\Sigma_{i=1}^{n} d_i = O(|E| + |V|)$, where E is the set of edges and V is the set of vertices, in the generated graph. Since $|E| > |V|$, complexity $= O(|E|)$

2.3 Generating a connected graph

We see that each one of the graphs in Fig. 2.1 and Fig. 2.2 of the preceding section have multiple edges between pairs of vertices, have cut-vertices and are disconnected. The next theorem shows how to modify the conditions to ensure the generation of a connected graph.

Theorem 2 *[7] The necessary and sufficient conditions for a set of integers $d_1, d_2, \ldots, d_n$ to be realizable as a connected graph are:*
(i) the set $d_1 \leq d_2 \leq d_3 \leq \cdots \leq d_n$ is realizable, that is, satisfies the conditions of Theorem 1
(ii) $\Sigma_{i=1}^{n} d_i \geq 2(n-1)$

2.3.1 Algorithm 2

The generation algorithm relies on the idea of edge-switching. We first generate a graph based on the Algorithm derived from Theorem 1. Assume this graph has $r > 1$ disjoint connected components. If none of these components have a cycle then the total number of edges in the graph is $\Sigma_{i=1}^{r} n_i - 1 = n - r$, where n_i is the number of vertices in the i-th component. Thus the total degree of the graph is $2(n-r)$ which contradicts the second condition of Theorem 2 as $r > 1$. Thus, at least one of the components contains a cycle. We can pick an edge of a cycle from such a component and an edge from another component (that may not have a cycle) and use edge-switching to merge the two components into one. This is shown in Fig. 2.3. Unless we are reduced to a single component, we can continue the process until there is only one connected component.

For an example, we observe that the graph in Fig. 2.1 has two disjoint components. We can merge the two components into one, by removing an edge between the vertices 2 and 3 and an edge between 0 and 5, replacing this pair of edges with an edge between 2

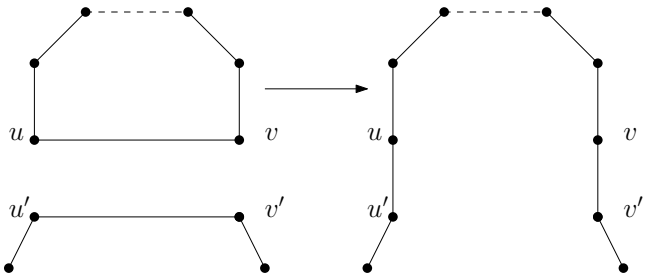

Figure 2.3: *Merging two components, one of which contains a cycle, by edge-switching*

and 5 and another between 3 and 0.

We describe the algorithm formally. Let $C =< C_1, C_2, \ldots, C_r >$ be the list of disjoint connected components in the graph G output by Algorithm 1. Search through the list of components C, to find a component C_i with a cycle. Let C_j, $i \neq j$, be any other component. Pick an edge e of a cycle in C_i and and any edge e' of the component C_j. By edge-switching, as explained above, merge the components C_i and C_j into a new component C_{ij} and let $C' = C - \{C_i, C_j\} \cup \{C_{ij}\}$ be the new list of disjoint components. Repeat the above for the new component list C', unless $|C'| = 1$, that is, only one connected component is left.

To find a component with a cycle, we iterate through the list C. Given a component C_i, we check if the number of edges of this component is equal to the number of its vertices minus 1, in that case, it is a tree and we move on to the next component in the list. If not, we search for a pair of vertices, connected by multiple edges. If such a pair exists, any pair of edges constitute a cycle and serves our purpose; else, we use depth-first search to find a cycle.

2.3.2 Some Implementation Details

To implement the above algorithm we maintain the following data structures:

1. An adjacency list for the graph G, with an additional field for each list node that records edge-multiplicity.

2. For each component of the graph we maintain a record of the "root" vertex that was used to visit all the other vertices of the component, a cyclic edge, if there are any cycles in the component, and a non-cyclic edge, if any.

After the initial depth first search, each subsequent depth first search merges a component with a cycle with another that may or may not contain a cycle. Two such components are picked by going through the component records, making sure that one of the components has a cyclic edge in its record. One of the edges is picked from each component record so that at least one of the two is cyclic and the two edges are switched by updating the adjacency information of the four end-points of the switched edges in

the adjacency list. We now do a partial depth-first search, starting at the "root" vertex of one of the components and create a record for the new component, arising out if the merger of the two components. The previous records for the merged components are deleted. The merging process is repeated until there is only one connected component left.

Algorithm 2 *connectedMultigraph*(v_i)

Input: Non-negative, non-increasing integer sequence
Output: Connected multigraph
G =*looplessMultigraph*(v_i)
components, multiedges = DFS(G)
while *length*(*components*) > 1 **do**
    Choose *component*1 with multiple edge $\{u, v\}$
    Choose *component*2 with edge $\{x, y\}$
    *component*1 = $\{u, v\}$ and $\{x, y\}$ with $\{u, x\}$ and $\{v, y\}$
    Remove *component*2
    *multiedge*1 = DFS(*component*1)
end while

2.3.3 Complexity

If the number of components in the graph output from the Algorithm based on Theorem 1 is equal to r. The complexity of producing a single connected component is $O(r(|V| + |E|))$.

2.4 Alternate Characterization

An equivalent characterization of Theorem 2 is given in following Corollary.

Corollary 1 *[7] The set of integers $d_1, d_2, \ldots, d_n$ is realizable as a connected graph if:*
(i) the set $d_1 \leq d_2 \leq d_3 \leq \ldots \leq d_n$ is realizable, and
(ii) $d_4 \neq 1$.

We show that the condition (ii) above implies condition (ii) of Theorem 2.

As $d_4 \neq 1$ and the d_i's are all positive integers, $d_4 \geq 2$. Since, $d_4 \leq d_5 \leq d_6 \leq \ldots$, $d_4 + d_5 + \ldots + d_n \geq 2(n-3)$ as there are $n-3$ terms in the sequence, each greater than or equal to 2. Thus $d_1 + d_2 + d_3 + d_4 + \ldots + d_n \geq 2(n-3) + 3$ (the 3 is for $d_1 + d_2 + d_3$).

Now the right-hand side of the inequality is odd and hence equality cannot hold. Therefore, LHS $> 2(n-3) + 3$ or $\geq 2(n-3) + 4 = 2(n-1)$ as required.

An interesting problem is to be able to use the second condition as a stand-alone criterion for generating a graph.

2.5 Generating a graph without a cut-vertex

Hakimi [7] established the following theorem. Note that a graph is non-separable means that it has no cut-vertex.

Theorem 3 *[7] The necessary and sufficient conditions for a set of integers $d_1, d_2, \ldots, d_n$ to be realizable as a non-separable graph are:*
(i) $\Sigma_{i=1}^{n} d_i = 2e$*, where e is an integer.*
(ii) $d_i \neq 1$ *for all i.*
(iii) $\Sigma_{i=1}^{n-1} d_i \geq d_n + 2(n-2)$

2.5.1 Algorithm 3

It is easily shown that the derived sequence $d' = (d_1 - 2, d_2 - 2, \ldots, d_n - 2)$ is realizable, satisfying all the conditions of Theorem 1. Thus we use Algorithm 1 to generate a graph realizing the sequence d'. We superimpose on this graph a Hamiltonian cycle passing through all the vertices, giving us a graph that is not separable.

Consider the degree sequence $d = (2, 2, 2, 2, 3, 3)$. It satisfies all the conditions of Theorem 3. However, the algorithm based on Theorem 1 generates the graph of Fig. 2.1. The graph satisfying the reduced degree sequence $d' = (0, 0, 0, 0, 1, 1)$ consists of 4 isolated vertices and a pair edges between the remaining two vertices.

Superposing a Hamiltonian cycle on all 6 vertices give us the non-separable, connected graph of Fig. 2.4.

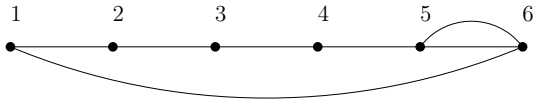

Figure 2.4: *A non-separable, connected graph realizing the sequence d = (2, 2, 2, 2, 3, 3)*

Algorithm 3 *connectedNonseparableMultigraph*(v_i)

Input: Non-negative, non-increasing integer sequence
Output: Connected non-separable multigraph
for $i = 1 \rightarrow length(v_i)$ **do**
    $v_i[i] = v_i[i] - 2$
end for
$G =$*looplessMultigraph*(v_i)
for $i = 1 \rightarrow length(v_i)$ **do**
    Add edge $\{v_i, v_{i+1}\}$
end for
Add edge $\{v_n, v_1\}$

2.5.2 Correctness

The degree of each vertex in the input graph is greater than or equal to 2. After subtracting 2 degrees from each vertex the graph constructed on the residual degrees is a disjoint graph if the degrees of some of the vertices is reduced to 0. The Hamiltonian cycle constructed on this graph will add the 2 subtracted degrees back to each vertex and since the cycle is through all the vertices, there will be no cut vertex present in the graph. Hence the resulting graph is non separable.

2.6 Generating a connected but separable graph

Hakimi [7] established the following theorem for a graph to be connected but separable.

Theorem 4 *[7] A necessary and sufficient condition for a set of integers $d_1, d_2, \ldots, d_n$ that is realizable as a connected graph (satisfies conditions of Theorem 2) to be realizable as a separable but connected graph is:*
(i) If $n = 3$, $d_1 + d_2 = d_3$.
(ii) If $n = 4$, there must exist among the integers a d_i and a d_j such that $d_i \neq d_j$.
(ii) If $n > 4$, there must exist among the set an integer $d_i \neq 2$

2.6.1 Algorithm 4

For a generation algorithm, we first deal with the base cases. For $n = 3$, we start with three vertices corresponding to the three degrees, adding $n_{13} = d_1$ edges between 1 and 3 and $n_{23} = d_2$ edges between 2 and 3. The main idea underlying the construction is to split the degree of a graph between two graphs and superimpose the vertices of split degree. For $n \geq 4$ we proceed by distinguishing two cases:

1. $d_1 = d_2 = \ldots = d_n$.
   Since $d_1 \neq 2$, set $d_1 = 2 + d_{12}$. Consider now the sequences $d' = (2, d_2, d_3)$ and $d'' = (d_{12}, d_4, \ldots, d_n)$. Both d' and d'' are realizable as connected graphs. Superposing the vertices corresponding to the vertex of degree 2 in d' and the vertex of degree d_{12} in d'' gives us a graph that is connected and has d_1 as a cut vertex.

2. $d_1 \neq d_n$.
   In this case, the sequences $d' = (d_1, d_1)$ and $d'' = (d_2, d_3, \ldots, d_n - d_1)$ are realizable as connected graphs, Superposing the vertices corresponding to a degree d_1 in the first graph and the vertex of degree $d_n - d_1$ in the second graph gives us a connected graph that has the vertex of degree d_n, obtained by superposition, as a separable vertex.

For an example, consider the degree sequence $d = (3, 3, 3, 3, 3, 3)$, satisfying the first case. The graphs obtained by splitting the degree of the first vertex correspond to the sequences $d' = (2, 3, 3)$ and $d'' = (1, 3, 3, 3)$.

Superposing the vertex 0 of the left and right figures we get a separable, connected graph realizing the sequence $d = (3, 3, 3, 3, 3, 3)$ with a cut-vertex at 0.

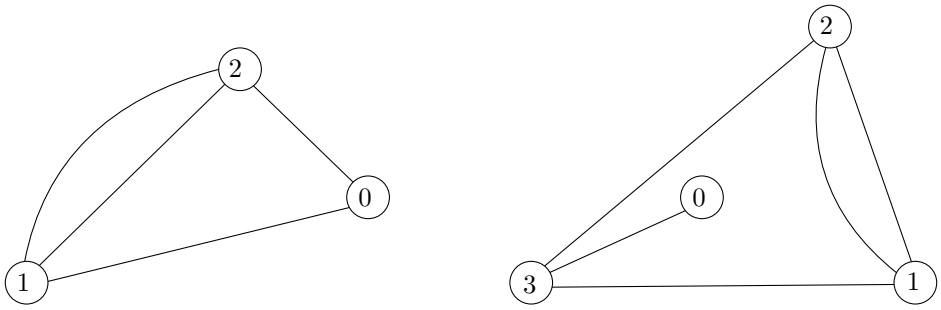

Figure 2.5: *(a) A non-separable, connected graph realizing the sequence d = (2, 3, 3) (b)A non-separable, connected graph realizing the sequence d = (1, 3, 3, 3)*

Algorithm 4 *connectedseparableMultigraph*(v_i)

Input: Non-negative, non-increasing integer sequence
Output: Connected separable multigraph
if $v_1 = v_n$ **then**
  Set $v1_i = 2, d_2, d_3$
  Set $v2_i = d_4, \ldots, d_{n-1}, d_n - 2$
  connectedMultigraph$(v1_i)$
  connectedMultigraph$(v2_i)$
  Overlap 2 from $v1_i$ to $d_n - 2$ vertex of $v2_i$
else
  Set $v1_i = d_1, d_1$

  Set $v2_i = d_2, \ldots, d_{n-1}, d_n - d_1$
  connectedMultigraph$(v1_i)$
  connectedMultigraph$(v2_i)$
  Overlap d_1 from $v1_i$ to $d_n - d_1$ vertex of $v2_i$
end if

2.6.2 Correctness

The graph degree is split into two subgraphs degrees on a vertex that is the cut vertex. Two connected graphs are generated using the two sets of degrees. These connected graphs might or might not have a cut vertex. However, once the two subgraphs are joined via the cut vertex the resulting graph is guaranteed to have a cut vertex that is the vertex whose degrees were split between the two sequences.

2.6.3 Complexity

The complexity of this algorithm is dominated by the construction of a connected graph for the longer sequence in each of the two cases and is therefore $O(r(|V| + |E|))$, where r is the number of components in the graph obtained by applying Algorithm 1 to the longer sequence.

Chapter 3

k-factors

3.1 Introduction

Let $d_G(v)$ denote the degree of a vertex v of a graph $G = (V, E)$. Given a map $f : V \to N \cup \{0\}$ where f is a function, an f-factor of G is a spanning subgraph H such that $d_H(v) = f(v)$ for each $v \in V$. The f-factor theorem of Tutte [14], gives necessary and sufficient conditions for the existence of an f-factor.

The degree sequence problem we discussed in Chapter 2 can be interpreted as a special case of Tutte's theorem: Given that $f = \langle f(v_1), f(v_2), \ldots, f(v_n)\rangle$, this sequence is graphical if there exists an f-factor of the complete graph K_n on the n vertices of G.

For a constant k, a k-factor is a k-regular spanning subgraph of G. A graphic degree sequence $(d_1, d_2, \ldots, d_n)$ is said to be k-factorable if there exists a realization that has a k-regular spanning graph as a subgraph.

In this chapter we focus on various aspects of k-factorable graphic degree sequences.

Rao and Rao [11] made the following conjecture:

Conjecture 1 *Given a constant k, a graphic degree sequence $(d_1, d_2, \ldots, d_n)$ is k-factorable if and only if the sequence $(d_1 - k, d_2 - k, \ldots, d_n - k)$ is graphic.*

Kundu [10] proved the slightly more general version of this conjecture:

Theorem 5 *Let $(d_1, d_2, \ldots, d_n)$ and $(d_1 - k, d_2 - k, \ldots, d_n - k)$ be two graphical sequences with the property that for some $k \geq 0$, $k \leq k_i \leq k + 1$ for all i. Then there exists a graph with degree sequence $(d_1, d_2, \ldots, d_n)$, containing a $(k_1, k_2, \ldots, k_n)$-factor.*

We first consider the problem of generating graphic degree sequences that are k-factorable. That requires us to solve, in the first place, the problem of generating graphic sequences.

We can try to solve the Erdos-Gallai Inequalities (EGI) to determine a sequence d that satisfies these. However, the *min*-function on the right-hand side of these inequalities is problematic. What comes to our rescue are a set of sufficient conditions in [15]

that make no direct reference to the EGI.

Choose integer parameters a and b such that $a \geq b > 0$, and let $K(a, b)$ denote the class of sequences d such that:

$$a \geq d_1 \geq d_2 \geq \ldots \geq d_n \geq b > 0 \tag{3.1}$$

The following result that shows each sequence in the class $K(a, b)$ to be graphic.

Theorem 6 *[15] Let $K(a, b)$ be a class of degrees such that $a \geq b > 0$. If degree sequence $d \in K(a, b)$ and $n = |d| \geq (a + b + 1)^2/4ab = l$, then d is graphic.*

An intuitive interpretation of Theorem 6 is that if we choose a sequence d longer l, then d is going to be graphic.

In the next section discuss an algorithm due to Chen [4] that gives a constructive method for generating a k-factorable graphic sequence by producing a k factor of a graphic realization. Interestingly, its implementation requires graphic sequences as input, which is why we discussed a method for doing this earlier. Other reasons for discussing this algorithm is that (a) it brings together in an interesting cocktail the problems of graphicality of degree sequences, packing degree sequences and k-factorability of degree sequences; (b) there seems to be no analysis available of the complexity of this algorithm.

3.2 Generating k-factorable degree sequences

Chen [4] gave a very simple and elegant proof of Kundu's result using graph packing technique. This proof also suggests an algorithm for the construction of a k-factor. We first explain the algorithm with the help of an example and then describe the algorithm formally.

Graph packing: Let $\alpha = (a_1, a_2, \ldots, a_n)$ and $\beta = (b_1, b_2, \ldots, b_n)$ be two graphic sequences. Simple graphs A and B that realizes the degree sequences α and β are said to pack if A and B are edge disjoint and $deg_A(v_i) = a_i, deg_B(v_i) = b_i$ [12].

This example is based on the algorithm suggested in Theorem 16 of Section 3.2 in Tyler Seacrest's thesis [12]. This theorem is based on Chen's proof. Below, we will follow the notations of this thesis.

Consider the graphic degree sequence: $\pi = (3, 3, 3, 3, 2, 2)$ so that $n = 6$. Let $k = 1$.

The graph A of Fig. 3.1 is an initial realization of the sequence: $\pi - k = \pi - 1 = (2, 2, 2, 2, 1, 1)$. while the graph B of Fig. 3.1 is an initial realization of the sequence: $n - 1 - \pi = 6 - 1 - \pi = (2, 2, 2, 2, 3, 3)$.

Superposition of the initial realizations of the graphs A and B is the multigraph C of Fig. 3.2.

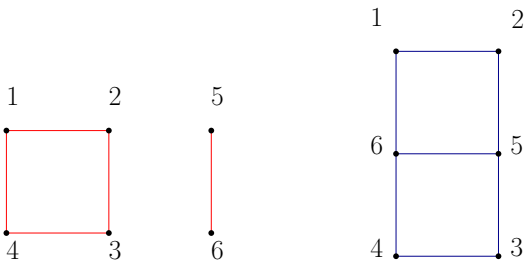

Figure 3.1: *Graphs A and B*

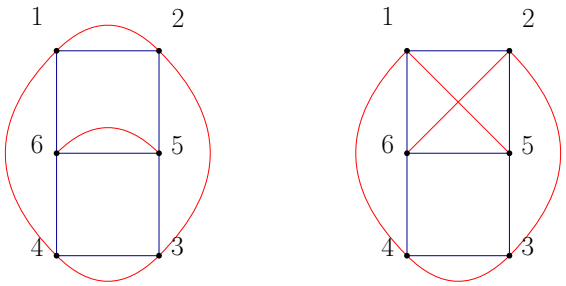

Figure 3.2: *Graphs C and D*

We remove one of the edges of the multi-edge between the vertices 5 and 6 from the graph B, by switching the edges $\{1, 2\}$ and $\{5, 6\}$ in this graph by adding the replacement edges $\{6, 2\}$ and $\{5, 1\}$ to the same graph. This also removes one of the edges in the multi-edge between 1 and 2 and gives us the graph D of Fig. 3.2.

Remark: Why should edge-switching be possible? Recall that the vertices involved in this edge-switching are u, v, x and y, where there is no edge between v and x, and there is a multi-edge between u and v. Let the total degree of x be expended by joining the set of vertices, say, V_x, each of which is connected to x by one or two edges. Note that v is not in V_x.

Now, since u is joined to v by two edges and x is not joined to v, the total degree of both u and x being the same, u cannot be joined to the vertices in X to match that of x. Thus there must a exist vertex y in X such that x is joined to y by one edge and there is no edge joining u to y, or x is joined to y by two edges, and u is joined to it by at most one edge.

In the first case, edge-switching removes one of the two edges between u and v. In the second case, one of the edges between each of the pairs $\{u, v\}$ and $\{x, y\}$ is removed and possibly an edge is introduced between u and y to make it a multi-edge. In each case, however, there is a net reduction of the total number of multi-edges.

Consider the graph C of Fig. 3.2. Let $u = 6$ and $v = 5$. Since the total degree of v is 4 with degrees used up bu the edges from the vertex u into it, the remaining 2 degrees are spread among vertices 1, 2, 3 and 4. Thus there exists a vertex it is not joined to. In this

case these are vertices 1 and 4. Set $x = 1$. The total degree 4 of 1 is distributed among the vertices $\{6, 2, 4\}$. This is V_x. Note that $v = 5$ is not in V_x. Since $u = 6$ is joined to $v = 5$ with 2 edges and is also joined to 1, it has only 1 more degree to expend on the vertices 2 and 4 in $V_x - \{6\}$. On the other hand, 1 has three more degrees to expend on the vertices 2 and 4. Hence there exists a vertex in $V_x - \{6\}$ that 6 is not connected to. In this case it is 2. □

We still have a multi-edge between 3 and 4 in the superposed graph. We switch the pair of edges $\{1, 2\}$ and $\{3, 4\}$ in the graph A with the replacement pair $\{1, 3\}$ and $\{2, 4\}$. This gives us the graph E of Fig. 3.3.

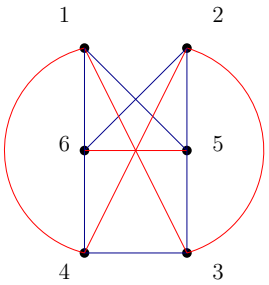

Figure 3.3: *Graph E*

The new realizations of the graphs B (figure on left) and B (figure on right) are now shown in Fig. 3.4.

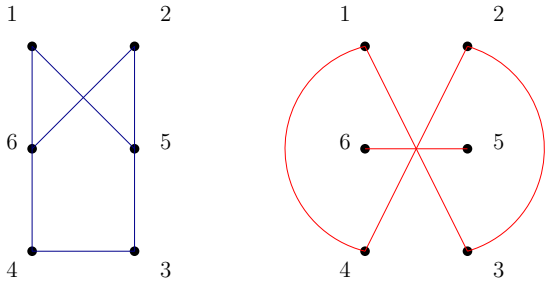

Figure 3.4: *New Graphs B and A*

The complement of the new graph B is a realization of $\pi = (3, 3, 3, 3, 2, 2)$. Both this graph (figure on left) and a 1-factor (figure on right), which is the graph $\overline{B} \backslash A$, are shown in Fig. 3.5.

We give a formal description of the algorithm below.

Algorithm *k-Factor*

Step 1. Compute a realization of a graph A that is a realization of the sequence $(d_1 - k, d_2 - k, \ldots, d_n - k)$.

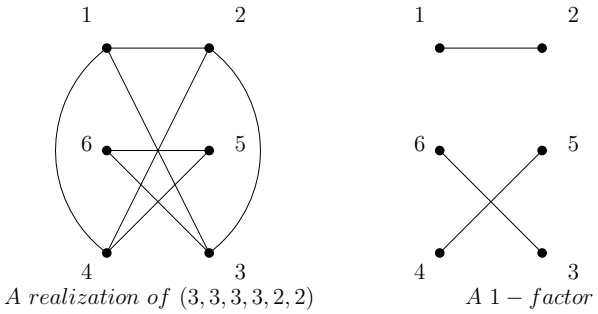

Figure 3.5: *A realization of (3, 3, 3, 3, 2, 2) and a 1-factor of this*

Step 2. Compute a realization of a graph B that is a realization of the sequence $(n-1-d_1, n-1-d_2, \ldots, n-1-d_n)$.

Step 3. Compute a superposition of the graphs A and B and make a list L of the multiple edges in the graph $A \cup B$.

Step 4. While there is a multi-edge $\{u, v\}$ in $A \cup B$ do:

Step 4.1. Find a vertex x that is not joined to v and a vertex w that is connected by at most one edge with u.

Step 4.2. Perform edge-switching by switching edges $\{u, v\}$ and $\{x, y\}$ with new edges between $\{v, x\}$ and $\{u, w\}$ in one of the graphs A or graph B.

We have implemented the above algorithm in Python. In the next section, we give an analysis of the complexity of this algorithm.

3.2.1 Complexity Analysis of Chen's Algorithm

Though an algorithm is implied by Theorem 16 of Seacrest's thesis, an analysis of the algorithm is missing. Such an analysis is useful for coming up with an efficient algorithm.

We first have to consider the complexity of finding overlapping edges in the superpositions of the graphs A and B. This can be done by taking a logical AND of the entries of the adjacency matrices of A and B and then scanning the entries of the resulting matrix for 1's.

The next step is to consider the complexity of removing multiple edges. For each multi-edge $\{u, v\}$, we have find vertices x and y to perform edge switching in one of the graphs A or B. Finding vertices x and y takes $O(n)$ time, while edge-switching can be done in $O(1)$ time.

After each multiple edge removal we might have to update the list of multiple edges as we might remove two multiple edges and add a new one.

This preliminary analysis seems to suggest that the complexity of this algorithm is $O(n^2 + mn)$, where m is the initial number of multiple edges found by the logical AND operation of the adjacency lists of the graphs A and B.

3.3 Generating factorable graphic sequences with connected k-factors

In [11] it is shown that a graphical degree sequence $(d_1, d_2, \ldots, d_n)$ with a k-factor has a connected k-factor if the following condition holds for $s < n/2$.

$$\Sigma_{i=1}^{s} d_i < s(n-s-1) + \Sigma_{i=0}^{s-1} d_{n-i} \tag{3.2}$$

The following trial-and-error heuristic can be used to generate a sequence that is graphic with a connected k-factor.

Step 1. Choose parameters a and b such that $a \geq b > 0$ and an integer $k \geq 2$.

Step 2. Choose a sequence satisfying Condition (3.1) that is longer than l as in Theorem 6.

Step 3. Verify that $d - k$ is also graphic. If not, pick another sequence.

Step 4. If both d and $d - k$ are graphic check if all the inequalities in Equation 3.2 are satisfied.

Step 5. If so, return this sequence, else go to to Step 2.

For $a = 10$ and $b = 3$, $n \geq 17$; set $k = 3$. The following sequence $d = (10, 10, 10, 10, 9, 9, 9, 9, 8, 8, 8, 8, 7, 7, 7, 7, 6, 4)$ has a connected 3-factor as seen in a realization, shown in Fig. 3.6.

3.3.1 Generalizing the result of Zverovich and Zvervich

For generating graphical sequences we have used the result of theorem 6. We use this result to pick graphical sequences that have connected k-factors.

From the inequality 3.2 we can say that a graph will have connected k-factors if it satisfies this inequality for each $s < n/2$. That is:

$\Sigma_{i=1}^{s} d_i < s(n-s-1) + \Sigma_{i=0}^{s-1} d_{n-i}$, where $s < n/2$

By maximizing the left hand side and minimizing the right hand side of the inequality we can say that

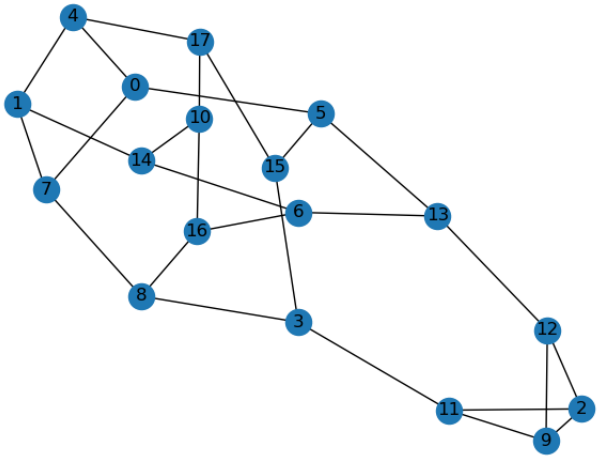

Figure 3.6: *Connected 3-factor for the degree sequence (10, 10, 10, 10, 9, 9, 9, 9, 8, 8, 8 , 8, 7, 7, 7, 7, 6, 4)*

$\Sigma_{i=1}^{s} d_i = sa$ and $\Sigma_{i=0}^{s-1} d_{n-i} = sb$

So, the inequality now becomes:

$sa < s(n-s-1) + sb$, or

$a - b < n - s - 1$

For further minimizing the right-hand side, we can maximize the value of s. Since $s < n/2$ set $s = n/2 - 1$. Plugging in this value of s we have:

$a - b < n/2$ or $n > (a-b)/2$

Using Theorem 6, we know that $n \geq (a+b+1)^2/4b$. Therefore,

Theorem 7 *For generating a graphical sequence with connected k-factor choose a sequence of length* $n > max(2(a-b), (a+b+1)^2/4b)$

3.4 Generating factorable graphic sequences with disconnected k-factor

We examine the problem of constructing examples of factorable graphic sequences with disconnected k-factors. In the following claim, we identify a class of such sequences for k=2.

Let $d = (n-1, ..., n-1, x, ...x, s, ..., s)$ be a sequence whose leading s terms are $n-1$, trailing s terms are s and all intermediate terms are x.

For an $s < n/2$, from the equation above we get:

$$(n-1) * s < s(n-s-1) + s * s,$$

which is false for all $s < n/2$.

Claim 1 *For n even and $5 \leq x \leq n-3$, the sequences $d = (n-1, n-1, x, ...x, 2, 2)$ are 2-factorable with disconnected factors.*

Proof: We show that both d and $d-2$ are graphic. We saturate the degrees of the first two vertices by joining them to each other and to all the remaining vertices. This reduces the degree sequence to $(x-2, x-2, ..., x-2)$ of length $n-4$.

Since n was assumed to be even, and $x \leq n-3$, we can construct an $x-2$-regular graph on $n-4$ vertices vertices, unless $x-2 = 1$, when we can construct a perfect matching.

The sequence $d-2 = (n-3, n-3, x-2, ..., x-2)$ is also graphic since there exists a graph on $n-2$ vertices, where each of the vertices of degree $n-3$ is each joined to the remaining $n-4$ vertices and to each other.

This reduces the degree of each of the trailing $n-4$ vertices by 2 and it is easy construct a regular graph on these vertices so that each is of degree $x-4$, unless $x-4 = 1$, when we construct a perfect matching.

Remark: We can reformulate and prove the above claim, stated in terms of a parameter $s \geq 2$, with $s+3 \leq x \leq n-5+s$ for sequences $d = (n-1, ..., n-1, x, x, ..., x, 2, 2, ..., 2)$, with $n-1$ as the first s terms, followed by $n-2s$ terms x, and s trailing 2's, bringing up the rear.

For $d = (15, 15, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 2, 2)$, the Fig. 3.7 and Fig. 3.8 below show graphic realizations of d and $d-2$, produced by a program that we wrote.

Are there other choices of s and the unspecified $n-2s$ intermediate values for which the sequences are graphic, with disconnected k factors for suitable choices of k ?

For example, by experimentation, using software we have developed, we found that for $n = 10$ and $s = 3$, the sequences $d = (9, 9, 9, 6, 6, 6, 6, 3, 3, 3)$, and $d = (9, 9, 9, 5, 5, 5, 5, 3, 3, 3)$ are graphic and have disconnected 2-factors.

In Fig. 3.9 and Fig. 3.10, respectively, a realization of the graphic sequence $d = (9, 9, 9, 6, 6, 6, 6, 3, 3, 3)$ and its 2-factor, consisting of two disjoint cycles, are shown.

Are there other choices of s and the unspecified $n-2s$ intermediate values for which the sequences are graphic, with disconnected k-factors for suitable choices of k?

For example, by experimentation, using software we have developed, we found that for $n = 10$ and $s = 3$, the sequences $d = (9, 9, 9, 6, 6, 6, 6, 3, 3, 3)$, and $d = (9, 9, 9, 5, 5, 5, 5, 3, 3,$

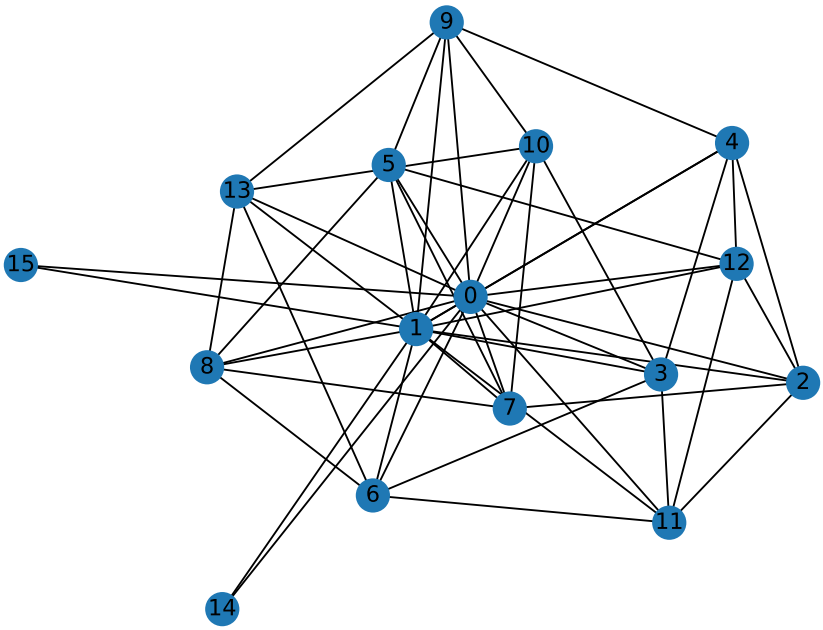

Figure 3.7: *Graph for the degree sequence (15, 15, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 2, 2)*

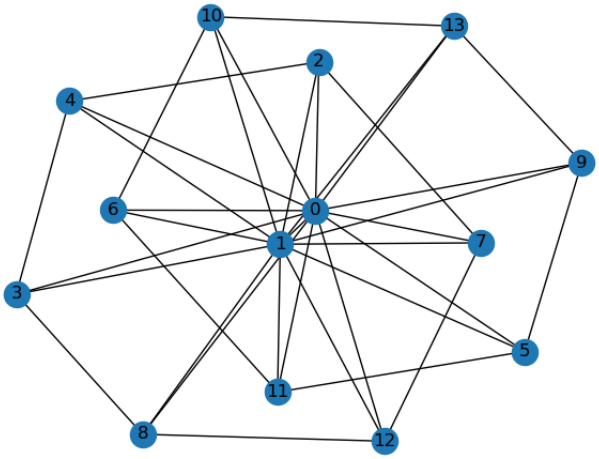

Figure 3.8: *Graph for the degree sequence (13, 13, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4)*

3) are graphic and have disconnected 2-factors.

It is interesting to note that the two disjoint cycles that make up the 2-factor cannot be merged into a single cycle. This is because of the structure of the subgraph C_1 on the vertices $\{0, 1, 2, 7, 8, 9\}$. Let $A_1 = \{7, 8, 9\}$ be the vertices at an even distance from 8, and $A_2 = \{0, 1.2\}$ at an odd distance from 8 in this subgraph. Then the vertices of A_1 are independent in the parent graph and the induced subgraph on vertices of A_2 is complete and each of its vertices is joined to all the vertices of the induced subgraph C_2 on the vertices $\{3, 4, 5, 6\}$. The notation $C_1 \to C_2$ is used to denote this relationship between C_1 and C_2.

Rao and Rao [11] proved the following interesting result.

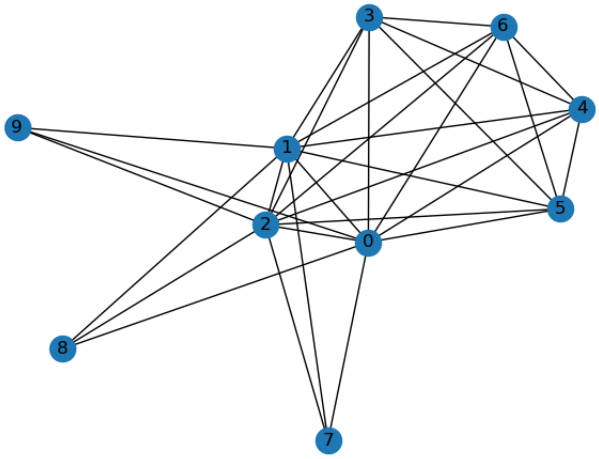

Figure 3.9: *Graph for the degree sequence (9, 9, 9, 6, 6, 6, 6, 6, 3, 3, 3)*

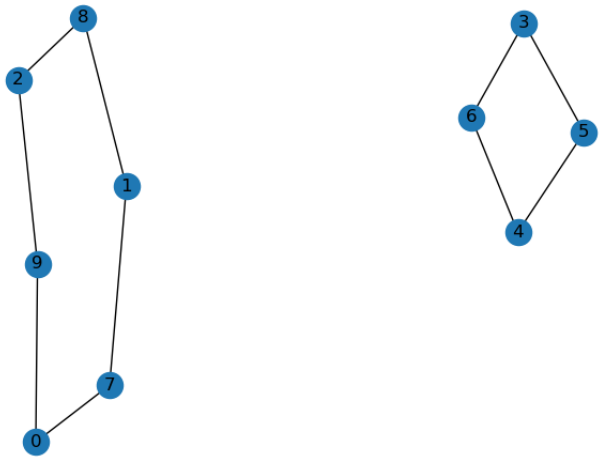

Figure 3.10: *2-factors of the graph for the degree sequence(9, 9, 9, 6, 6, 6, 6, 6, 3, 3, 3)*

Theorem 8 *Let G be a graph with a k-factor F consisting of two components C_1 and C_2. Let $k \geq 2$ and let C_1 and C_2 be bi-coherent. If the degree sequence of G is not connected k-factorable then either $C_1 \to C_2$, or $C_2 \to C_1$.*

Chapter 4

Directed graphs

4.1 Introduction

For a set of non-negative integer pairs of the form $d = [(d_i^+, d_i^-) : i = 1, 2, \ldots, n]$ in lexicographically decreasing order, does there exist a graph $G = (V, E)$ with vertices $(d_1^+, d_1^-), (d_2^+, d_2^-), \ldots, (d_n^+, d_n^-)$, where d_i^+ is the outdegree and d_i^- is the indegree for a vertex v_i of G. If there exists a graph G corresponding to the degree sequence, then d is said to be graphic and G is the realization of d.

In this section, we look at a theorem by Hakimi [8] that give necessary and sufficient conditions for a non-negative set of integer pairs to be graphic. For these conditions, we propose a generation algorithm whose correctness is ensured by sufficient conditions.

Following the algorithm for graph realization based on Hakimi's conditions for graphicality of a sequence, we will see how to prove the sufficiency of the theorem by Fulkerson-Ryser [6].

4.2 Generating directed graphs based on Hakimi's conditions

Let $d =< (d_1^+, d_1^-), (d_2^+, d_2^-), \ldots, (d_n^+, d_n^-) >$ be a sequence of positive pairs of integers, such that the sequence is in lexicographically decreasing order.

Theorem 9 *The set of pairs of positive integers $(d_i^+, d_i^-), \forall 1 \leq i \leq n$ can be realized as the directed graph if and only if:*
(i) $\Sigma_{i=1}^{n} d_i^+ = \Sigma_{i=1}^{n} d_i^-$
(ii) $\Sigma_{i=1}^{n-1}(d_i^+ + d_i^-) \geq d_n^+ + d_n^-$

4.2.1 Algorithm

This algorithm works on degree sequences in lexicographically decreasing order. We take the highest integer pair and reduce its outdegree one by one by adding an edge starting from the next highest in lexicographic order and reducing their indegree. Once the

outdegree of the highest pair is saturated, the sequence is rearranged to be in decreasing order again.

Rearranging the sequence ensures that the condition $\Sigma_{i=1}^{n-1}(d_i^+ + d_i^-) \geq d_n^+ + d_n^-$ is always satisfied. The algorithm ends when the outdegree of the highest pair is zero, and the outdegrees have been saturated. Due to the condition $\Sigma_{i=1}^{n} d_i^+ = \Sigma_{i=1}^{n} d_i^-$, the indegrees also have been saturated.

Algorithm 5 *directedGraphRealization*(v_i)

Input: Non-negative integer pair sequence(lexicographically decreasing order): $v_i = [(ov_i, iv_i)]$
Output: Directed graph
while $v_1[0] > 0$ **do**
  for $i = 2 \rightarrow v_1[0]$ **do**
    Add edge $v_1 \rightarrow v_i$
  end for
  Rearrange vertices in lexicographically decreasing order.
end while

4.2.2 Complexity

The complexity of the algorithm is proportional to the size of the graph that is generated. This is bounded above by $\Sigma_{i=1}^{n} d_i + n^2 = O(|E| + n^2)$, where E is the set of edges in the generated graph and n^2 is the time taken for rearranging the vertices after saturation.

4.3 A constructive proof of the Fulkerson-Ryser characterization of digraphic sequences

Given a sequence of nonnegative integer pairs of the form $d = \langle (od_i, id_i) \rangle$, $i = 1, 2, \ldots, n$ in lexicographically decreasing order, does there exist a digraph $G = (V, E)$ whose degree sequence is $(od_1, id_1), (id_2, id_2), \ldots, (od_n, id_n)$, where od_i is the outdegree and id_i is the indegree, respectively, of the vertex v_i of G.

If there exists such a graph G then d is said to be digraphic and G is a realization of the sequence d.

Tripathi et al. [13] gave a short constructive proof of the Erdos-Gallai characterization of graphic sequences [5]. In this section, we show that an analogous construction can be given for digraphic sequences based on the Ryser-Fulkerson characterization of such (see, for example, [3]).

Theorem 10 *A sequence* $(id_1, od_1), (id_2, od_2), \ldots, (id_n, od_n)$ *of nonnegative integers in non-increasing lexicographical order is realizable as a digraph if and only if:*

$$id_i \leq n-1, od_i \leq n-1, \forall i\ 1 \leq i \leq n$$
$$\Sigma_{i=1}^{n} id_i = \Sigma_{i=1}^{n} od_i$$
$$\Sigma_{i=1}^{r} od_i \leq \Sigma_{i=1}^{r} min\{r-1, id_i\} + \Sigma_{i=r+1}^{n} min\{r, id_i\}, \forall r\ 1 \leq r < n$$

4.4 Digraph Construction

The necessity of the inequalities is strightforward. The total outdegree ot the vertices in the *leftSet* can be accounted for by directed edges going from a vertex in the *leftSet* to a vertex in the *rightSet* (second term on the right-hand side of the inequality) and by outgoing edges from a vetex in the *leftSet* to another in the same set (first term on the right-hand side of the inequality).

The role of the *min* function is also easy to understand. For example, we have $min\{r, id_i\}$ as the a maximum of r edges can originate from vertices in the *leftSet* and upto a maximum of its indegree id_i. Hence we will have the minimum of the two values.

The problem is to construct a directed graph (digraph) G on n vertices $v_1, v_2, \ldots, v_n$ such that v_i has outdegree od_i and indegree id_i. We construct G incrementally, starting with an empty graph on n vertices. The outdegree and indegree of a vertex v_i at an intermediate stage will be denoted by $od(v_i)$ and $id(v_i)$ respectively. Clearly, $od(v_i) \leq od_i$ and $id(v_i) \leq id_i$ always holds.

We maintain three sets, S_1, S_2 and S_3 of degree pairs $(od(v_i), id(v_i))$. In the first set S_1 all outdegrees and indegrees are fully saturated, that is $od(v_i) = od_i$ and $id(v_i) = id_i$ for each i in this set; in the second set, S_2, the outdegrees are but not the indegrees, that is $od(v_i) = od_i$ and $id(v_i) \leq id_i$ for each i ; in the third set, S_3, the outdegrees are fully unsaturated and the indegrees partially saturated, that is $od(v_i) = 0$ and $id(v_i) \leq id_i$ for each i. In each set, the degree pairs are lexicographically ordered. Below, the term *leftSet* (respectively *rightSet*) will stand for $S_1 \cup S_2$ (S_3). Furthermore, if v_i and v_j are two vertices in S_3, there no edge connecting them.

Let r be the largest index such that for all i in $1 \leq i < r$, $od(v_i) = od_i$. We add outgoing edges to v_r till $od(v_r) = od_r$, when v_r is said to be saturated. When adding edges, several cases arise as discussed below.

Case 0: If there exists an i such that $v_r \not\rightarrow v_i$ and $id(v_i) < id_i$ we add the edge $v_r \rightarrow v_i$.

In Case 0, we explored vertices v_i for which either $od(v_i) = 0$ or the vertices for which $od(v_i) = od_i$ but $id(v_i) < id_i$. That is, we explored vertices in the right and middle sets. If v_r could not be joined to v_i then that is because of two possibilities:

(a) v_r is already joined to v_i and $id(v_i) \leq id_i$.

(b) v_r cannot be joined to v_i as $id(v_i) = id_i$.

Assume one of the possibilities (a) or (b) is true. Now, we have to look to the left set, that is, vertices for which $id(v_i) = id_i$ and $od(v_i) = od_i$. There are two cases to consider:

Case 1: There exist a $v_i \in leftSet$ such that $v_i \not\to v_r$.

This case can be divided into two cases.

Case 1.1 $id_r - id(v_r) > 0$: in this case, we find an edge $v_r \to u$ such that $u \notin N(v_r)$ and replace it with edges $\{v_i \to v_r, v_r \to u\}$, as shown in Fig. 4.1.

The existence of such an u follows from the fact that $od(v_i) = od_i \geq od_r > od(v_r)$.

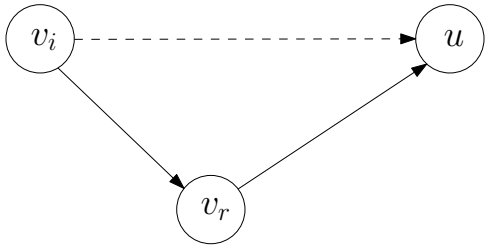

Figure 4.1: *Rewiring Step 1(a)*

Case 1.2 $id_r - id(v_r) = 0$ and $id_r > 0$: in this case, we find edges $\{v_i \to u, v_k \to v_r\}$ and replace them with the edges $\{v_i \to v_r, v_r \to u\}$, as shown in Fig. 4.2.

The existence of v_k follows from the fact that indegree of v_r is saturated; since there is no outgoing edge from any of the vertices in the $rightSet$, there has to be a vertex $v_k \neq v_i$ such that $v_k \to v_r$.

The existence of u such that there is an edge $v_i \to u$ but $v_r \not\to u$ follows from the same inequalities as in Case 1.1.

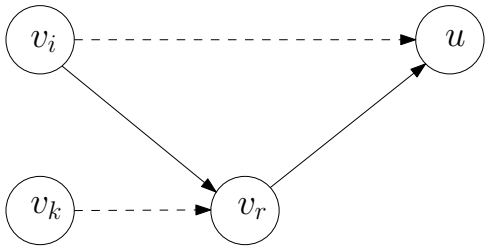

Figure 4.2: *Rewiring Step 1(b)*

Case 2: There exists $k > r$ such that $id(v_k) < min\{r, id_k\}$.

The condition $id(v_k) < min\{r, id_k\}$ ensures that v_k is not saturated. There exists a vertex v_i in the left set such that $v_i \not\to v_k$ as it cannot be v_r. There also exists vertex u for which $id(u) = id_u$ and $v_r \not\to u$. This is guaranteed by the inequalities invoked in Case 1.1 and which still hold.

We now replace $v_i \to u$ with $\{v_r \to u, v_i \to v_k\}$ as in Fig. 4.3.

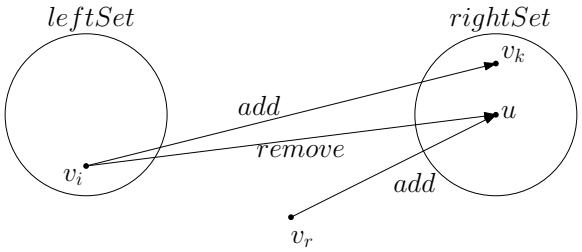

Figure 4.3: *Rewiring in Case 2*

Case 3: For each vertex v_i in the $leftSet$, we have $v_i \to v_r$, and the condition $id(v_j) < min\{k-1, id_j\}$ for some j ensures that there exists v_i such that $v_i \not\to v_j$.

We argue as before that there exists u in the right set such that $v_i \to u$ but $v_r \not\to u$. We then replace $v_i \to u$ with $\{v_r \to u, v_i \to v_j\}$.

The rewiring is schematically shown in Fig. 4.4.

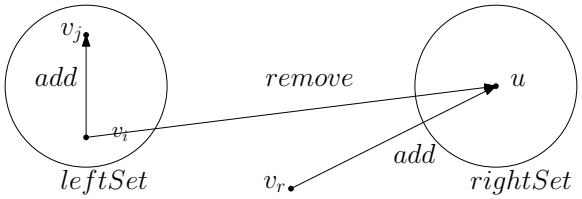

Figure 4.4: *Rewiring in Case 3*

If none of the above cases hold, then $id(v_i) = min(r-1, id_i)$ for $i = 1, 2, \ldots, r$ and $id(v_i) = min(r, id_i)$ for $i = r+1, r+2, \ldots, n$. Now $\Sigma_{i=1}^{n} id(v_i) = \Sigma_{i=1}^{n} od(v_i)$ is an invariant of the construction process. Then from the Ryser-Fulkerson inequalities, it follows that $\Sigma_{i=1}^{r} od_i \leq \Sigma_{i=1}^{r} od(v_i)$. However, $od(v_i) \leq od_i$ for each $i = 1, \ldots, r$, which implies that $\Sigma_{i=1}^{r} od_i \geq \Sigma_{i=1}^{r} od(v_i)$. Hence $\Sigma_{i=1}^{r} od_i = \Sigma_{i=1}^{r} od(v_i)$. From the choice of r, it follows that the first $r-1$ terms on either side of the equality are equal and therefore $od(v_r) = od_r$, Since v_r is outdegree-saturated we add it to the *leftset*, increment r by 1, and continue.

Algorithm 6 *fulkersonRyserRealization*(v_i)

Input: Non-negative integer pair sequence(lexicographically decreasing order): $v_i = [(ov_i, iv_i)]$
Output: Directed graph
$S_1, S_2 = []$
$S_3 = v_i$
while $length(S_3) > 0$ **do**
    $r_i =$ pop $S_3[0]$
    Call fulkersonRyserCase0()
    Call fulkersonRyserCase1()
    Call fulkersonRyserCase2()
    Call fulkersonRyserCase3()
end while

Algorithm 7 *fulkersonRyserCase0*

Input: S_1, S_2, S_3, r_i
for each item in S_3 **do**
    if $or_i = 0$ **then**
        break
    end if
    if S_3 has v_i where $iv_i > 0$ and $r_i \not\rightarrow v_i$ **then** Add $r_i \rightarrow v_i$
    end if
end for
for each item in $S_1 \cup S_2$ **do**
    if $or_i = 0$ **then**
        break
    end if
    if S_1 or S_2 has v_i where $iv_i > 0$ and $r_i \not\rightarrow v_i$ **then**
        Add $r_i \rightarrow v_i$
    end if
    $or_i := or_i - 1$
end for

Algorithm 8 *fulkersonRyserCase1*(v_i)

Input: S_1, S_2, S_3, r_i
while $or_i > 0$ **do**
  if $or_i = 0$ **then**
    break
  end if
  if S_1 or S_2 has v_i and S_3 has u and $ir_i > 0$ and $r_i \not\to u$ and $v_i \to u$ **then**
    Remove $v_i \to u$
    Add $v_i \to r_i$ and $r_i \to u$
  else if S_1 or S_2 has v_i, v_k and S_3 has u and $ir_i = 0$, $v_k \to r_i$, $v_i \to u$, and $r_i \not\to u$ **then**
    Remove $v_i \to u$ and $v_k \to r_i$
    Add $v_i \to r_i$ and $r_i \to u$
  end if
end while

Algorithm 9 *fulkersonRyserCase2*(v_i)

Input: S_1, S_2, S_3, r_i
while $or_i > 0$ **do**
  if $or_i = 0$ **then**
    break
  end if
  if S_1 or S_2 has v_i and S_3 has v_k, u and $v_i \to u$, $v_i \not\to v_k$ and $r_i \not\to u$ **then**
    Remove $v_i \to u$
    Add $v_i \to v_k$ and $r_i \to u$
  end if
end while

Algorithm 10 *fulkersonRyserCase3*(v_i)

Input: S_1, S_2, S_3, r_i
while $or_i > 0$ **do**
  if $or_i = 0$ **then**
    break
  end if
  if S_1 or S_2 has v_i, v_j and S_3 has u and $v_i \to u$, $v_r \not\to u$ and $v_i \to u$ **then**
    Remove $v_i \to u$
    Add $v_i \to v_j$ and $v_r \to u$
  end if
end while

Chapter 5

Conclusion

This thesis provides a comprehensive review of various criteria that can determine the graphicality of a sequence. We begin by going through some criteria for the sequence to be realized as an undirected graph with vertex degrees corresponding to the integer sequences, then prove their sufficiency by generating algorithms capable of constructing graphs based on the integer by treating them as degree sequences.

We discussed necessary and sufficient conditions for the integer sequences to be realized as the undirected graph with specific properties, as stated by Hakimi. We have proposed algorithms to generate graphs whose vertex degrees are a given sequence of positive integers $d =< d_1, d_2, \ldots, d_n >$, under different sets of sufficient conditions. Each set of sufficient conditions guarantee that the generated graph satisfy some property (properties).

Degree sequences that satisfy specific conditions can have k-factors. We look at the conditions stated by Kundu for detecting such sequences and implement Seacrest's algorithm for realizing graphs and their k-factors for a given degree sequence. We analyze the complexity of the algorithm to measure its efficiency. Using the result by Rao and Rao to detect if the sequence can have realizations with connected k-factors, we generate a pattern that can construct sequences that do not have realizations with connected k-factors. Further, we use Zverovich's result to create a condition to ensure that the sequence always has connected k-factors.

We then take the necessary and sufficient conditions given by Hakimi and Ryser-Fulkerson characterization for realizing the integer pair sequence as directed graphs and generate algorithms for realizing graphs and proving the sufficiency of the theorems discussed.

All the algorithms have been implemented in Python 3.

5.1 Future works

Section 3.4 specifies conditions for which the sequence will have disconnected k-factors. An open problem is to use these conditions to generate integer sequences that can be

realized as a degree sequence of a graph.

The k-factors discussed in this thesis focuses on the undirected graphs and the conditions to generate sequences apply to the undirected graphs. One open problem is to use these conditions and extend them to realize directed graphs and its k-factors.

Bibliography

[1] A. Arman, P. Gao, and N. Wormald. "Fast uniform generation of random graphs with given degree sequences". In: *Random Structures & Algorithms* 59.3 (2021), pp. 291–314.

[2] E. A. Bender and E. R. Canfield. "The asymptotic number of labeled graphs with given degree sequences". In: *Journal of Combinatorial Theory, Series A* 24.3 (1978), pp. 296–307. ISSN: 0097-3165. DOI: https://doi.org/10.1016/0097-3165(78)90059-6. URL: https://www.sciencedirect.com/science/article/pii/0097316578900596.

[3] A. Berger. "A note on the characterization of digraphic sequences". In: *Discrete Mathematics* 314 (2014), pp. 38–41. ISSN: 0012-365X. DOI: https://doi.org/10.1016/j.disc.2013.09.010. URL: https://www.sciencedirect.com/science/article/pii/S0012365X13004068.

[4] Y. C. Chen. "A short proof of Kundu's k-factor theorem". In: *Discrete Mathematics* 71.2 (1988), pp. 177–179. ISSN: 0012-365X. DOI: https://doi.org/10.1016/0012-365X(88)90070-2. URL: https://www.sciencedirect.com/science/article/pii/0012365X88900702.

[5] P. Erdős and T. Gallai. "Graphs with prescribed degrees of vertices". In: *Mat. Lapok* 11.264-274 (1960), p. 15.

[6] D. R. Fulkerson. "Zero-one matrices with zero trace." In: *Pacific Journal of Mathematics* 10.3 (1960), pp. 831–836.

[7] S. L. Hakimi. "On Realizability of a Set of Integers as Degrees of the Vertices of a Linear Graph. I". In: *Journal of the Society for Industrial and Applied Mathematics* 10.3 (1962), pp. 496–506. DOI: 10.1137/0110037. eprint: https://doi.org/10.1137/0110037. URL: https://doi.org/10.1137/0110037.

[8] S. L. Hakimi. "On the degrees of the vertices of a directed graph". In: *Journal of the Franklin Institute* 279.4 (1965), pp. 290–308. ISSN: 0016-0032. DOI: https://doi.org/10.1016/0016-0032(65)90340-6. URL: https://www.sciencedirect.com/science/article/pii/0016003265903406.

[9] M. Koren. "Sequences with a unique realization by simple graphs". In: *Journal of Combinatorial Theory, Series B* 21.3 (1976), pp. 235–244. ISSN: 0095-8956. DOI: https://doi.org/10.1016/S0095-8956(76)80007-X. URL: https://www.sciencedirect.com/science/article/pii/S009589567680007X.

[10] S. Kundu. "The k-factor conjecture is true". In: *Discrete Mathematics* 6.4 (1973), pp. 367–376. ISSN: 0012-365X. DOI: https://doi.org/10.1016/0012-365X(73)90068-X. URL: https://www.sciencedirect.com/science/article/pii/0012365X7390068X.

[11] A. R. Rao and S. B. Rao. "On factorable degree sequences". In: *Journal of Combinatorial Theory, Series B* 13.3 (1972), pp. 185–191. ISSN: 0095-8956. DOI: https://doi.org/10.1016/0095-8956(72)90055-X. URL: https://www.sciencedirect.com/science/article/pii/009589567290055X.

[12] T. Seacrest. "Packing and Realizations of Degree Sequences with Specified Substructures". PhD thesis. University of Nebraska, 2011.

[13] A. Tripathi, S. Venugopalan, and D. B. West. "A short constructive proof of the Erdős–Gallai characterization of graphic lists". In: *Discrete Mathematics* 310.4 (2010), pp. 843–844. ISSN: 0012-365X. DOI: https://doi.org/10.1016/j.disc.2009.09.023. URL: https://www.sciencedirect.com/science/article/pii/S0012365X09004683.

[14] W. T. Tutte. "A Short Proof of the Factor Theorem for Finite Graphs". In: *Canadian Journal of Mathematics* 6 (1954), pp. 347–352. DOI: 10.4153/CJM-1954-033-3.

[15] I. E. Zverovich and V. E. Zverovich. "Contributions to the theory of graphic sequences". In: *Discret. Math.* 105.1-3 (1992), pp. 293–303. DOI: 10.1016/0012-365X(92)90152-6. URL: https://doi.org/10.1016/0012-365X(92)90152-6.

Vita Auctoris

Name: Daniel John
Place of birth: Kerala, India
Year of birth: 1993
Education: Bachelor of Technology in Computer Science and Engineering from Cochin University of Science and Technology
Master of Science in Computer Science from University of Windsor, Windsor, Canada (Fall 2021- Winter 2023)