ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ
Τμημα Μαθηματικων

Δημήτριος Βρόσγος

Πραγματοποιηση Γραφηματων υπο περιορισμους στους Βαθμους και στις Αποστασεις

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΑΤΡΙΒΗ

Ιωάννινα, 2024

UNIVERSITY OF IOANNINA

Department of Mathematics

Dimitrios Vrosgos

Graph Realization under Degree and Distance Constraints

Master's Thesis

Ioannina, 2024

Εις μνήμην του πατέρα μου.

Η παρούσα Μεταπτυχιακή Διατριβή εκπονήθηκε στο πλαίσιο των σπουδών για την απόκτηση του Μεταπτυχιακού Διπλώματος Ειδίκευσης στα Εφαρμοσμένα Μαθηματικά και Πληροφορικής, που απονέμει το Τμήμα Μαθηματικών του Πανεπιστημίου Ιωαννίνων.

Εγκρίθηκε την 09/07/2024 από την εξεταστική επιτροπή:

Ονοματεπώνυμο	**Βαθμίδα**
Χάρης Παπαδόπουλος	Αναπληρωτής Καθηγητής
Μιχαήλ Μπέκος	Επίκουρος Καθηγητής
Λουκάς Γεωργιάδης	Αναπληρωτής Καθηγητής

ΥΠΕΥΘΥΝΗ ΔΗΛΩΣΗ

"Δηλώνω υπεύθυνα ότι η παρούσα διατριβή εκπονήθηκε κάτω από τους διεθνείς ηθικούς και ακαδημαϊκούς κανόνες δεοντολογίας και προστασίας της πνευματικής ιδιοκτησίας. Σύμφωνα με τους κανόνες αυτούς, δεν έχω προβεί σε ιδιοποίηση ξένου επιστημονικού έργου και έχω πλήρως αναφέρει τις πηγές που χρησιμοποίησα στην εργασία αυτή."

Δημήτριος Βρόσγος

Ευχαριστιες

Με την ολοκλήρωση της μεταπτυχιακής διατριβής μου, θα ήθελα να απευθύνω ολοψύχως ένα ευχαριστώ σε όλους όσους στάθηκαν δίπλα μου και με τον δικό τους τρόπο με βοήθησαν να τα καταφέρω.

Ιδιαίτερα θα ήθελα να ευχαριστήσω τον επιβλέποντα μου κ. Χάρη Παπαδόπουλο, Αναπληρωτή Καθηγητή του τμήματος Μαθηματικών, του Πανεπιστήμιο Ιωαννίνων, για την συνεχή υποστήριξη και καθοδήγησή του.

Τέλος, θα ήθελα να ευχαριστήσω την μητέρα μου και τον αδερφό μου, που ήταν πάντα στο πλευρό μου και με βοήθησαν σε ό,τι και αν χρειάστηκα.

Περιληψη

Στην παρούσα εργασία, εξετάζουμε την ύπαρξη ενός γραφήματος με βάση διαφορετικούς περιορισμούς στους βαθμούς και στις αποστάσεις μεταξύ των κορυφών. Αυτό που μας ενδιαφέρει κυρίως είναι να μελετήσουμε την υπολογιστική πολυπλοκότητα του κάθε προβλήματος και να εξετάσουμε περιπτώσεις όπου το πρόβλημα λύνεται σε πολυωνυμικό χρόνο ή αποδεικνύεται NP-πληρότητα.

Τα προβλήματα που μελετάμε κατά βάση σε αυτή την εργασία αποτελούν παραλαγές ενός πολύ γνωστού προβλήματος στην θεωρία γραφημάτων. Δοθέντος μιας ακολουθίας βαθμών $(d_1, \ldots, d_n)$, το πρόβλημα μας ζητάει να αποδείξουμε την ύπαρξη ή μη ένος γραφήματος που να ικανοποιεί την ακολουθία. Είναι γνωστό ότι το θεμελειώδες αυτό πρόβλημα επιδέχεται πολυωνυμικό αλγόριθμο. Οι παραλαγές που εξετάζουμε σχετίζονται με τους βαθμούς των κορύφων. Για παράδειγμα βλέπουμε το πρόβλημα πραγματοποίησης ενός απλού γραφήματος με $v_1, \ldots, v_n$ κορυφές, και πρέπει να ελέγξουμε εάν υπάρχει γράφημα τέτοιο ώστε για κάθε $i \in [1, n]$, ο μέγιστος βαθμός στην γειτονία της v_i να είναι d_i. Ένα διαφορετικό, αλλά αντίστοιχης λογικής, πρόβλημα είναι όταν ελέγχουμε τον ελάχιστο βαθμό στην γειτονία μιας κορυφής.

Τέλος, αυτό που διαφέρει από τις προηγούμενες περιπτώσεις είναι το πρόβλημα πραγματοποίησης με βάση την απόσταση των κορυφών. Η διαφορά εδώ είναι ότι δεν έχουμε μια ακολουθία βαθμών, αλλά έναν ολόκληρο $n \times n$ πίνακα μη αρνητικών ακεραίων που αντιπρωσοπεύουν επιθυμητές αποστάσεις μεταξύ κορυφών, και πάλι σκοπός είναι να βρούμε ένα γράφημα που να ικανοποιεί τους περιορισμούς που περιγράφονται από τον πίνακα.

Σε όλες τις παραλλαγές των προβλημάτων παραθέτουμε και αναλύουμε την υπολογιστική πολυπλοκότητα και που υπάρχει στη πρόσφατη βιβλιογραφία. Αναδεικνύουμε διαφορετικούς περιορισμούς που τα προβλήματα παρουσιάζουν υπολογιστική διχοτόμιση και παρουσιάζουμε αλγορίθμους που έχουν προταθεί για να αντιμετωπιστούν.

Abstract

In this paper, we consider the existence of a graph under different constraints on degrees and distances between vertices. What we are mainly interested in, is to study the computational complexity of each problem and to consider cases where the problem is solved in polynomial time or is proved to be NP-complete.

The problems we basically study in this paper are variations of a well-known problem in graph theory. Given a degree sequence $(d_1, \ldots, d_n)$, the problem asks us to prove the existence or non-existence of a graph satisfying the sequence. This fundamental problem is known to be polynomial-time solvable. The variations we consider are related to the degrees of the vertices. For example we see the problem of realizing a simple graph with $v_1, \ldots, v_n$ vertices, and we need to check if there exists a graph such that for each $i \in [1, n]$, the maximum degree in the neighborhood of v_i to be d_i. A different but similar problem is when we check the minimum degree in the neighborhood of a vertex.

Finally, what differs from the previous cases is the realization problem based on vertex distance. The difference here is that we don't have a sequence of degrees, but a whole $n \times n$ matrix of non negative integers, and again the goal is to find a graph that satisfies the matrix.

In all variations of the considered problems we present and analyze their computational complexity that has been suggested in the recent related literature. We expose different restrictions under which the problems admit computational dichotomy and we present algorithms that have been proposed to deal with such constraints.

Contents

CHAPTER 1

Introduction

1.1 Graph theory

In mathematics, graph theory is the study of graphs, which are mathematical structures used to model pairwise relations between objects. A graph in this context is made up of vertices (also called nodes or points) which are connected by edges (also called arcs, links or lines).

The main definition of a graph is given below.

Definition 1.1. *A graph is an ordered pair $G = (V, E)$, where V is a set of vertices (also called nodes or points) and $E \subseteq \{\{x, y\} | x, y \in V, x \neq y\}$ is a set of edges (also called links or lines), which are unordered pairs of vertices (that is, an edge is associated with two distinct vertices).*

In general, graph theory is used to solve problems that can be represented by a graph. One of the oldest and best-known problems in graph theory is the seven Bridges of Königsberg problem which was published in 1736 by Leonard Euler and actually was the first paper in the history of graph theory. Since then, everyday problems have been modeled in graphs in order to be solved afterwards. Graph theory is now used in many sciences, such as chemistry, physics or biology. Graph theory is also widely used in sociology as a way, for example, to measure actors' prestige or to explore rumor spreading, notably through the use of social network analysis software. Under the umbrella of social networks are many different types of graphs. Acquaintanceship and friendship graphs describe whether people know each other. Influence graphs model whether certain people can influence the behavior of others. Finally, collaboration graphs model whether two people work together in a particular way, such as acting in a movie together, etc.

Definition 1.2. *Two vertices, let x and y, of a graph G are called adjacent* *when the edge that joins them exists.*

The edge of a graph is denoted by $e = (x, y)$, where x and y are the vertices it connects and are called endpoints. The edges can be directed (asymmetric) or undirected (symmetric). A distinction is made between undirected graphs, where edges link two vertices symmetrically, and directed graphs, where edges link two vertices asymmetrically. The definitions are given for greater precision.

Definition 1.3. *A graph $G = (V, E)$ is called directed if the set of edges E is ordered, i.e. the edges have an orientation.*

Definition 1.4. *A graph $G = (V, E)$ is called undirected if the edges have no orientation. The edge (a, b) is identical to the edge (b, a), that is, there are no ordered pairs.*

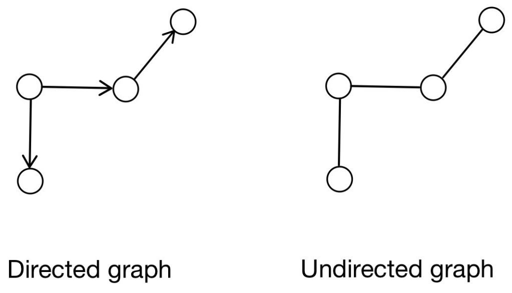


Figure 1.1: Example of a directed graph and an undirected graph

1.2 Computational complexity

The Complexity Theory is a key building block of algorithm analysis and central domain of computer science. Historically, the English mathematician Alan Turing is considered the "father of computer science", thanks to his enormous contribution to the field of the theory of computation during the 1930s, but also or artificial intelligence, thanks to the test Turing, which he proposed in 1950. One way to experimentally determine whether one machine has authentic cognitive abilities and can think. He finally gave the previously informal concept of an algorithm a formal, mathematical formulation through the Turing Machine.

As we said in the previous section, graph theory is used to solve problems that can be represented by a graph. To solve these problems it is important to have efficient algorithms. In computer science, the computational complexity or simply complexity of an algorithm is the amount of resources required to run it. Particular focus is given to computation time (generally measured by the number of needed elementary operations) and memory storage requirements. The complexity of a problem is the complexity of the best algorithms that allow solving the problem.

Definition 1.5. *In mathematics and computer science, an algorithm is a finite sequence of rigorous instructions, typically used to solve a class of specific problems or to perform a computation.*

We are mainly interested in the efficiency of the algorithm in terms of its execution time. More simply we are interested in algorithms running fast.

Definition 1.6. *The computational complexity or running time of a deterministic turing machine M is the function $f : \mathbb{N} \to \mathbb{N}$, where $f(n)$ is the maximum number of steps that M can take, when given an input size n.*

So the running time directly depends on the number of iterations of the algorithm and on the size of the input. According to the asymptotic analysis to calculate the computational complexity the fastest time is polynomial for each size of the input $O(n^c)$, for some c. In the diagram below we can see different forms of algorithm running times, noting that an algorithm is efficient when it has polynomial time and $O(1)$ is the best, since has constant time.

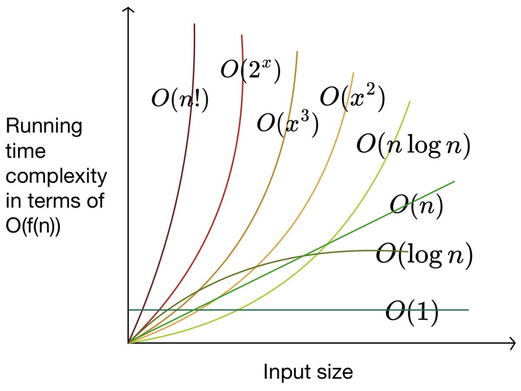


Figure 1.2: Running time complexity

1.3 Graph realization problem

In order to be able to study the problem, it is necessary to give some useful definitions that exist in graph theory. So we have that the degree of a vertex of a graph is the number of edges that are incident to the vertex. The degree of a vertex v is denoted $deg(v)$ or $degv$. The maximum degree of a graph G denoted by $D(G)$, and the minimum degree of a graph, denoted by $d(G)$, are the maximum and minimum of its vertices' degrees. The degree sequence of a graph is a non-increasing degree sequence and is denoted $(d_1, \ldots, d_n)$, where $d_1 > \cdots > d_n$.

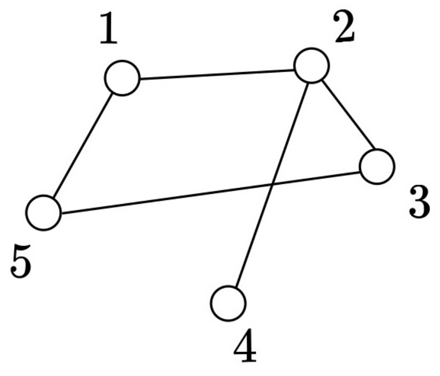


Figure 1.3: Vertex degrees example

In Figure 1.3 we can see an example of a simple, undirected graph G, in which holds that $deg(1) = 2, deg(2) = 3, deg(3) = 2, deg(4) = 1$ and $deg(5) = 2$. This means that the maximum degree of the graph is 3, i.e. $D(G) = deg(2) = 3$. Respectively, the minimum degree of the graph is 1, i.e. $d(G) = deg(4) = 1$.

One of the oldest problems in graph theory is the Degree Sequence Realizability, where it is a decision problem.This problem has received a great deal of attention in recent years, due to its importance in applications such as wireless sensor networks and structural biology. Given a degree sequence $(d_1, \dots, d_n)$ of positive integers, the problem asks whether there is an undirected simple graph such that $(d_1, \dots, d_n)$ is the degree sequence of this graph. A decision problem is a computational problem that can be posed as a yes–no question and therefore in this case, we are asked to answer whether there is one or more graphs that have the sequence of degrees given in the problem.

The problem can be solved in polynomial time. One method to show this uses the Havel-Hakimi algorithm [23, 26], constructing a special solution with the use of a recursive algorithm. Alternatively, following the characterization given by the Erdös-Gallai theorem [19], the problem can be solved by testing the validity of n inequalities.

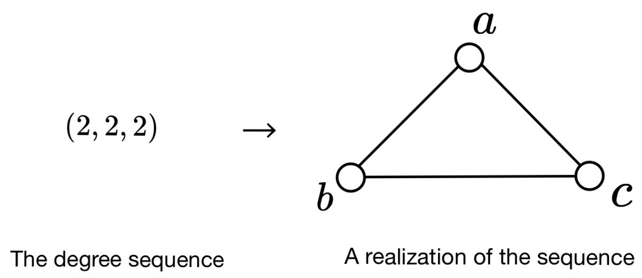


Figure 1.4: Example of graph realization problem

An example is given in Figure 1.4 for convenience. Given a sequence of positive integer degrees, we need to decide whether there is a simple undirected graph that has this degree sequence. In the example there is indeed such a graph as shown in the figure. The data we have from the sequence is that we must have 3 vertices with degree 2. This graph is the complete K_3 graph. Obviously there are also sequences that we cannot represent with a graph. If we had, for example, the sequence $(4, 3, 2)$, then there would be no graph, since the sequence requires that we have only 3 vertices and one of them must have degree 4. This is impossible since there are not that many vertices to connect with the one that needs degree 4.

CHAPTER 2

Extensions of Graph Realization Problem

While the realizability problem of degree sequences has been studied for different classes of graphs, there has been relatively work concerning the realizability of others types of information profiles, such as the vertex neighborhood profiles. There are now many extensions and variations of this problem, such as restrictions on degrees and distances between vertices. More specifically, we have studied problems where we choose the maximum degree in vertex neighborhood or the minimum degree and problems where we have a whole $n \times n$ matrix, instead of a sequence of degrees. This paper focuses on studying these problems.

2.1 Graph Realization of Distance Sets

In this chapter we study a different case of graph realization, called Distance realization problem and defined as follows.

Definition 2.1. *Given a $n \times n$ matrix D of non negative integers, find an n-vertex weighted or unweighted graph G realizing D, i.e., whose inter vertex distances satisfy $dist_G(i,j) = D_{i,j}$ for every $1 \leq i < j \leq n$ (where $dist_G(i,j)$ is the distance of vertices i and j in G), or decide that no such realizing graph exists.*

What is the difference between weighted and unweighted graph?

- A weighted graph is a graph in which a number (the weight) is assigned to each edge. Such weights might represent for example costs, lengths or capacities, depending on the problem at hand.

- In an unweighted graph, all edges are equal and there's no specific cost associated with moving from one vertex to another.

The problem was studied for general weighted and unweighted graphs, as well as for cases where the realizing graph is restricted to a specific family of graphs such as trees, paths, bipartite graphs, cycles and others. Even in this problem there are extensions and variations that have been studied. More specifically when each entry in the matrix D can contain a range of allowed values. We refer to this extension as Range Distance Realization (or Range-DR). Limiting each range to at most k values results the problem k-Range Distance Realization (or k-Range-DR). One more extension of Distance Realization is when each entry $D_{i,j}$ of the matrix may contain an arbitrary set of acceptable values for the distance between i and j, for every $1 \leq i < j \leq n$. We refer to this extension as Set Distance Realization (or Set-DR). Limiting each entry to at most k values, the problem k-Set Distance Realization (or k-Set-DR) arises. [11]

The difference with the previous problems is that here the given profile is an $n \times n$ matrix D such that each entry $D_{i,j} \in \mathbb{N} \cup \{\infty\}$, for $1 \leq i < j \leq n$, and $D_{i,i} = \{0\}$ for every $1 \leq i \leq n$. A graph $G = (V, E)$ is a realization of D if $dist_G(i,j) = D_{i,j}$, for every $1 \leq i < j \leq n$, where $dist_G(i,j)$ denotes the distance between i and j in G. Here, we are interested in two types of realizing graphs, the unweighted Distance Realization and the weighted Distance Realization.

Observe that an unweighted realizing graph is fully determined by D. The edge (i,j) exists in the graph if and only if $D_{i,j} = 1$. This means that every edge of the realizing graph is of length 1. On the contrary in weighted Distance Realization the edges of the realizing graph may have any positive integral lengths.

2.2 Maximum Neighborhood Degree Realization Problem

In this section we study a special case of information profiles, concerning neighborhood degree. Such profiles are of theoretical interest in context of social networks. The natural problem in this direction concern the maximum degrees in the vertex neighborhoods. More specifically for each vertex i, let d_i denote the maximum vertex degree in i's neighborhood. So we have that

$\textsc{MaxNDeg}(G) = (d_1, \ldots, d_n)$ is the maximum neighborhood degree profile of G.

As before, so here too, we are concerned with the following central question.

Question 1. *Given a sequence $D = (d_1, \ldots, d_n)$ of non-negative integers, is there a simple graph with vertices $v_1, \ldots, v_n$ such that for every $i \in [1, n]$, the maximum degree in the neighborhood of v_i is exactly d_i?*

To be able to study this problem and for simplicity, we will represent the input vector D in a more compact form as $\sigma = (d_l^{n_l}, \ldots, d_1^{n_1})$, where n_i is a non-negative integer for each $1 \leq i \leq l$ and

$$\sum_{i=1}^{l} n_i = n$$

The graph G must contain exactly n_i vertices whose maximum degree in their neighborhood is d_i. Moreover, we assume that $d_l > d_{l-1} > \cdots > d_1 \geq 1$. Note that d_1 cannot be zero because then we would have n_1 vertices with maximum neighborhood degree equal to zero, i.e. we would have n_1 isolated vertices, but then the graph would not be connected and we would treat them separately.

There are two scenarios we study regarding the neighborhood of vertices.

- The first is the scenario of the closed neighborhoods, wherein a vertex belongs to its own neighborhood.
- The second is the scenario of the open neighborhoods, wherein a vertex is not counted in its own neighborhood.

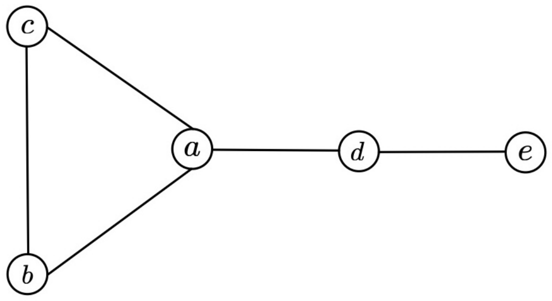


Figure 2.1: Max-neighboring-degree realization of profile:(i) $\sigma = (3^4, 2^1)$ with respect to closed neighborhood, and (ii) $\sigma = (3^3, 2^2)$ with respect to open neighborhood.

The graph in the Figure 2.1 depicts a MaxNDeg realization of $(3^4, 2^1)$ as MaxNDeg(a), MaxNDeg(b), MaxNDeg(c) and MaxNDeg(d) are all equal to $deg(a) = 3$ and MaxNDeg(e) $= deg(d) = 2$. However, it is a MaxNDeg$^-$ realization of $(3^3, 2^2)$ because MaxNDeg$^-$(b), MaxNDeg$^-$(c) and MaxNDeg$^-$(d) are all equal to $deg(a)$ which is 3 and MaxNDeg$^-$(a) $=$ MaxNDeg$^-$(e) $= deg(d) = 2$.

For greater convenience and understanding we will now give some definitions, which we will also need in our work.

2.2.1 Definitions and Notation

Let H be a simple undirected graph. We define all of the following.

- Denote by $V(H)$ (respectively, $E(H)$) the vertex (respectively, edge) set of H.
- For a vertex $v \in V(H)$, let $deg_H(v)$ denote the degree of v in H.
- For a set $W \subseteq V(H)$, denote by $H[W]$ the subgraph of H induced by the vertices of W.
- For a set $W \subseteq V(H)$ and a vertex $v \in V(H)$, denote by $W \cup v$ and $W \setminus v$, respectively, the sets $W \cup \{v\}$ and $W \setminus \{v\}$.

- Let $N_H(v) = \{u|(v,u) \in E(H)\}$ be the open neighborhood of v in H and let $N_H[v] = \{v\} \cup \{u|(v,u) \in E(H)\}$ be the closed neighborhood of v in H.
- For a vertex $v \in H$, let $\textsc{MaxNDeg}_H(v)$ denote the maximum degree among all the vertices in the closed neighborhood of v, and let $\textsc{MaxNDeg}_H{}^{-}(v)$ denote the maximum degree among all the vertices in the open neighborhood of v.

We are now in a position to formally define the concept of realizable profiles.

Definition 2.2. *Let $\sigma = (d_l^{n_l}, \ldots, d_1^{n_1})$ be a profile satisfying $d_l > d_{l-1} > \cdots > d_1 > 0$ and let $n = n_1 + \cdots + n_l$.*

Definition 2.3. *The profile σ is said to be* MaxNDeg *realizable if there exists an n-vertex graph G that for each $i \in [1, l]$ contains exactly n_i vertices whose* MaxNDeg *is d_i. Equivalently, $|\{v \in V(G) : \textsc{MaxNDeg}(v) = d_i\}| = n_i$.*

Definition 2.4. *The profile σ is said to be* MaxNDeg$^-$ *realizable if there exists an n-vertex graph G that for each $i \in [1, l]$ contains exactly n_i vertices whose* MaxNDeg$^-$ *is d_i. Equivalently, $|\{v \in V(G) : \textsc{MaxNDeg}^-(v) = d_i\}| = n_i$.*

2.3 Minimum Neighborhood Degree Realization Problem

In this unit we introduce and study the minimum degree in vertex neighborhood profile as it is one of the most common extensions of the classical degree profile to vertex neighboring degree profiles. Given a graph $G = (V, E)$, the min-degree of a vertex $v \in V$, namely $\textsc{MinNDeg}(v)$, is given by $min\{deg(w)|w \in N[v]\}$. Our input is again a sequence $\sigma = (d_l^{n_l}, \ldots, d_1^{n_1})$, where $d_{i+1} > d_i$ and each n_i is a non-negative integer. For each vertex i, we again define d_i as the minimum vertex degree in i's neighborhood. The same realizability questions asked above for degree sequences can be posed for minimum neighborhood degree profiles as well. This brings us to the following central question of our work.

Question 2. *Given a sequence $D = (d_1, \ldots, d_n)$ of positive integers. Is there a graph G of size n such that the minimum degree in the neighborhood of the i-th vertex in G is exactly equal to d_i?*

As in the maximum neighborhood degree, here too we will denote the input D in a more compact form as $\sigma = (d_l^{n_l}, \ldots, d_1^{n_1})$, where n_i's are positive integers with $\sum_{i=1}^{l} n_i = n$. The profile σ is said to be $\textsc{MinNDeg}$ realizable if there exists an n-vertex graph G such that $|\{v \in V(G) : \textsc{MinNDeg}(v) = d_i\}| = n_i$, namely, G contains exactly n_i vertices whose $\textsc{MinNDeg}$ is d_i, for every $i \in [1, l]$.

We again define two types of neighborhood, the closed neighborhood and the open neighborhood, with the difference that now we choose the minimum degree of the vertices in the neighborhood. An example is given for better understanding.

But first we note that as before, for the maximum degree, so here, the same definitions and notations apply respectively.

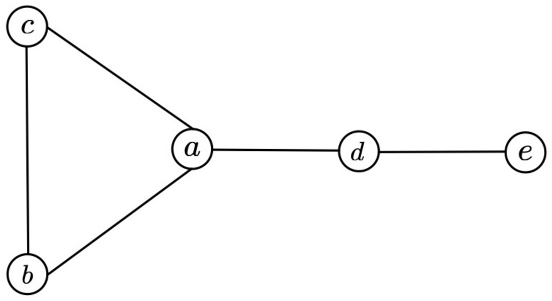


Figure 2.2: Min-neighboring-degree realization of profile:(i) $\sigma = (2^3, 1^2)$ with respect to closed neighborhood, and (ii) $\sigma = (2^4, 1^1)$ with respect to open neighborhood.

The graph in the Figure 2.2 depicts a MinNDeg realization of $(2^3, 1^2)$ as MinNDeg(a), MinNDeg(b) and MinNDeg(c) are all equal to $deg(b) = 2$ and MinNDeg(d) = MinNDeg(e) = $deg(e) = 1$. However, it is a MinNDeg$^-$ realization of $(2^4, 1^1)$ because MinNDeg$^-(b)$, MinNDeg$^-(c)$, MinNDeg$^-(a)$ and MinNDeg$^-(e)$ are all equal to $deg(c)$ which is 2 and MinNDeg$^-(d)$ = $deg(e) = 1$.

2.4 Related Work

Variations of these problems have been studied over the years, so it makes sense that there are many works related to this work in the literature. So in this section we mention a few similar works.

2.4.1 Graph Realization of Distance Sets

For example, an optimization variant of Distance Realization problem was introduced in [24]. In this problem, a distance matrix D is given over a set S of n vertices and the goal is to find a graph G including S, with possibly auxiliary vertices, that realizes the given distance matrix of S. It is also shown in [18] that an optimal realization can have at most n^4 vertices, and therefore there is a finite time algorithm to find an optimal realization. In [3] it is shown that finding optimal realizations of distance matrices with integral entries is

NP-complete.

Over the years, various heuristics for optimal realizations were considered [31, 33]. Since optimal realization seems hard even to approximate, special cases and other variations have been studied[21, 16]. Also special attention has been given to the optimal distance realization problem where the realizing graph is a tree. An $O(n^2)$ time algorithm for optimal tree-realization is described in [17].

2.4.2 Maximum and Minimum Neighborhood Degree Realization Problems

For both problems there is a large number of related questions, were considered in the literature, e.g., finding all the non -isomorphic graphs that realize a given degree sequence D, counting all non-isomorphic realizing graphs of D, sampling a random realization of D as uniformly as possible and determining the conditions under which a given D defines a unique realizing graph[10]. These realization questions are well-studied. Also degree realization with given constraints on some vertex was studied in [27]. For parallel and distributed realizations of degree sequences see [4, 5].

Graphic sequences were studied also on specific graph families, such as trees (see [22]), bipartite graphs [14, 34], planar graphs [1, 30], split graphs [25, 32], chordal, interval and perfect graphs [15].

The MaxNDeg and the MinNDeg realization problems have not been explored so far. Neighborhood lists and neighborhood degree lists were considered in [2, 13]. In these profiles, each vertex i is associated with the list of degrees of all vertices in its neighborhood. A related problem is the *shotgun assembly* problem [28], where the characteristic associated with the vertex i is some description of its neighborhood up to radius r. Several other realization problems are mentioned in [8, 9].

CHAPTER 3

Graph Realization of Distance Sets

In this chapter we study a well-known problem in graph theory, called graph realization of distance sets. We are interested in studying two types of realizing graphs. More specifically the unweighted and weighted graphs. In unweighted Distance Realization it is assumed that each edge of the realizing graphs is of length 1. In weighted Distance Realization the edges of the realizing graph may have any positive integer lengths.

Observe that an unweighted realizing graph is fully determined by D, because the edge (i, j) exists in the graph if and only if $D_{i,j} = 1$. Is is obvious that there is only one graph that can be the realizing graph. This was observed by Hakimi and Yau [24], who provided a characterization for distance realization by unweighted graphs, implying also a polynomial-time algorithm for unweighted DR.

Hakimi and Yau [24] also studied weighted DR. They proved that the necessary and sufficient condition for the realizability of a given matrix D is that D is a metric. Moreover, they gave a polynomial time algorithm that computes a realization for any given metric distance matrix. More specifically, their algorithm constructs a minimum edge realizing graph whose edges are necessary in every realization of D.

A natural extension of Distance Realization is when each entry in the distance matrix may contain a range of consecutive values instead of a single value. Tamura et al. obtained necessary and sufficient conditions for the realizability of a range distance matrix by weighted graphs, generalizing the results of [24] from precise to range specifications. A polynomial time algorithm for weighted Range-DR was given in [29]. The unweighted version of Range-DR was shown to be NP-hard in [12], where it was also shown that if the realizing graph is required to be a tree, then both the unweighted and weighted versions of Range-DR are NP-hard.

One of the main questions studied in this chapter involves the effect of limiting the number of values in each entry of the matrix D. This question is equally interesting for Set-DR and Range-DR. Given an integer k, we say that the matrix D is a k-set distance matrix if $|D_{i,j}| \leq k$ for every $1 \leq i < j \leq n$. A distance matrix D is a k-range distance matrix if $D_{i,j}$ is a range that contains at most k consecutive values for every $1 \leq i < j \leq n$.

In this chapter we study the computational complexity of k-Set-DR and k-Range-DR, as a function of k, in various graph families.

First of all, in [12] it is proved the hardness results for Range-DR by trees and unweighted graphs reveals that 3-Range-DR is NP-hard over these graph families, implying that 3-Set-DR is NP-hard as well. We modify the reductions from [12] to show that 2-Range-DR is NP-hard for general unweighted graphs, where precise realization is known to be polynomial [24]. For general weighted graphs, it is already known that Range-DR is computationally easy[29]. On the hardness side, we show that Set-DR is NP-hard for weighted stars already with 6-set distance profiles. We obtain slightly tighter results for paths and cycles, for which both unweighted and weighted Set-DR are NP-hard already with 5-set distance profiles. Our hardness results are based on reduction from the $3-$ Colorability problem [11]. However, the reductions are not similar. More specifically, in the case of weighted stars, the possible colors of a vertex are encoded in the distance matrix by possible edge weights. Also, in the case of weighted and unweighted paths, the colors are encoded by vertices and their location on the path. Finally, the hardness result for 5-Set-DR on the cycle is obtained by a reduction from 5-Set-DR on the path.[11]

Definition 3.1. *Suppose we have a graph G with n vertices, where there is at most one edge between any two vertices. The goal is to color the graph so that no two adjacent vertices are colored with the same color.*

We also show that 2-Set-DR is polynomial time solvable for stars and paths. Our realization algorithms are based on a reduction of the 2-SAT problem, which can be solved in linear time[20]. The idea is to use one vertex i_0 as a point of reference for all other vertices. A Boolean variable b_j is associated with each vertex j and determines which of the two values of $D_{i_0,j}$ should be used. The 2-SAT problem is constructed according to the rest of the entries of D. Applying this approach for stars is rather straightforward, but it becomes more complicated for paths.

3.1 Realizations by Unweighted General Graphs

For general graphs, recall that the range realization problem in weighted graphs has a polynomial time algorithm. We show an NP-hardness result for Range-DR by unweighted graphs, even for 2-range distance profiles.

Theorem 3.1. *[11] 2-Range-DR is NP-hard in unweighted general graphs*

Proof. To be able to prove this theorem, we will use a reduction from the 3-coloring problem.

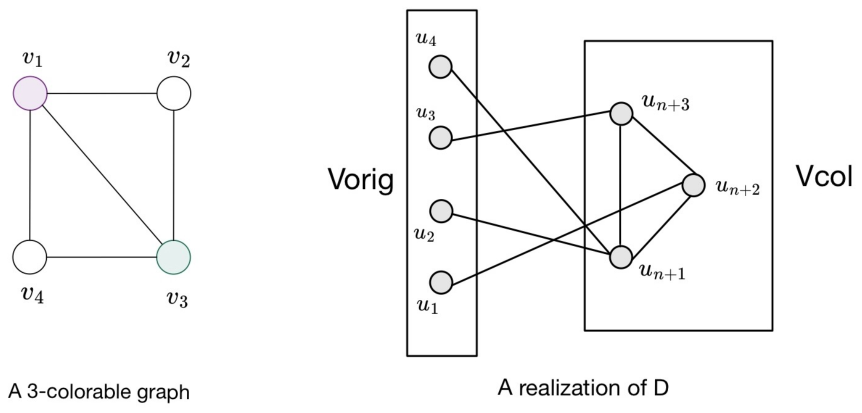


Figure 3.1: An example of the reduction for $n = 4$. White, purple and green correspond to nodes $n + 1, n + 2$ and $n + 3$, respectively

To begin with, we will consider a random graph G that is 3-colorable and then we will try to construct a 2-range distance matrix D for $n+3$ vertices, i.e. for the vertex set $\{u_1, \dots, u_{n+3}\}$. For greater convenience, we will consider for the initial n vertices the set $V_{orig} = \{u_1, \dots, u_n\}$, as representing the original vertices of the given graph G, and for the 3 additional vertices we will consider the set $V_{col} = \{u_{n+1}, u_{n+2}, u_{n+3}\}$, as representing the three colors. So the matrix can take the following values.

$$D_{i,j} = \begin{cases} \{1\}, & i = n+1, \dots, n+3, \ j = n+1, \dots, n+3 \\ \{1, 2\}, & i = 1, \dots, n, \ j = n+1, \dots, n+3 \\ \{2, 3\}, & 1 \le i < j \le n, \ (v_i, v_j) \notin E(G) \\ \{3\}, & 1 \le i < j \le n, \ (v_i, v_j) \in E(G) \end{cases}$$

Now we will show that the input G is 3-colorable if and only if D is realized by an unweighted graph.

We first assume that G is 3-colorable. So we define the coloring function as follows, $x : V(G) \mapsto \{1, \ldots, 3\}$. For the matrix D defined by G, we will construct a realizing graph G' as follows. We will start by creating a triangle containing the color vertices i.e., $u_{n+1}, u_{n+2}, u_{n+3}$. Connect each original vertex u_i to the color vertex $u_{n+x(i)}$. It is easily proved that G' realizes D (see Figure for an example).

For the other direction, we assume that there is an unweighted graph G' which realizes the matrix D. For every two original vertices u_i and u_j, since $1 \notin D_{i,j}$, it follows that these two vertices u_i and u_j are not connected by an edge. Therefore, certainly, every original vertex u_i must be connected to at least one of the three colored vertices. We define the color function of G as follows. For every original vertex u_i, let $n + c$, i.e. v_{col}, be some color vertex connected to u_i, and also let $x(v_i) = c$. Since $1, 2 \notin D_{i,j}$, if any two vertices v_i and v_j are connected by an edge in G, then their distance in G' must be at least 3. This makes us sure that none of the color vertices are connected to both u_i and u_j (as this would make their distance 2). It follows that if $(v_i, v_j) \in E(G)$, then v_i and v_j are assigned different colors.

□

In the next theorem we generalize the idea of theorem 3.1 from the 3-coloring to the k-coloring and prove the NP-hardness of the Range-DR in unweighted graphs. Also, it is proved that 2-Set-DR is NP-hard in unweighted and weighted trees[11].

Theorem 3.2. *[12] The unweighted distance-range realization problem DIST([ ],U) is NP-hard.*

Proof. We will prove that the problem is NP-hard using a reduction from the coloring problem.

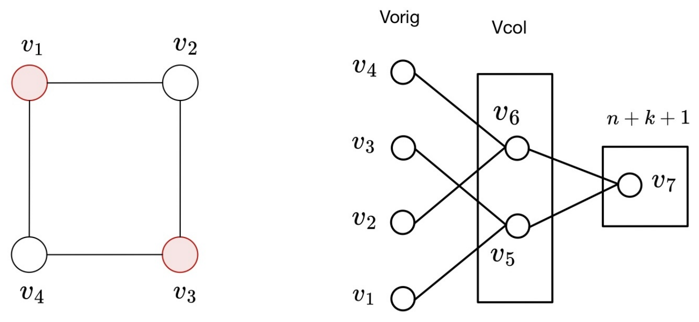


Figure 3.2: An example of the reduction for $n = 4$. White, purple and green correspond to nodes $n+1, n+2$ and $n+3$, respectively.

Consider an instance (G, k) of the coloring problem, namely, an unweighted undirected graph G and an integer k, where it is required to decide whether G can be colored with k or fewer colors. Let D be a distance matrix for $n + k + 1$ vertices, i.e., $\{1, \ldots, n + k + 1\}$. We define the first n vertices of D $V_{orig} = \{1, \ldots, n\}$, as representing the original n vertices of the graph G, and also we define the k vertices of D, $V_{col} = \{n + 1, \ldots, n + k\}$, as representing the colors, and we defined the last vertex $n + k + 1$ as coordinator.

First we will set some requirements on the colored vertices and the coordinator. More specifically, let $D_{n+l,n+k+1} = 1$ (i.e., all colored vertices with the coordinator) for every $1 \leq l \leq k$ and $D_{n+l,n+r} = 2$ (i.e., all the color vertices between them) for every $1 \leq l < r \leq k$. This allows only one possible realization for the subgraph induced by the vertices $V_{col} \cup \{n + k + 1\}$, i.e., a star rooted at $n + k + 1$ and there are no edges between the leaves.

Then for the connection between the original vertices and the star, we define the distance constraints as follows. Let $D_{i,n+k+1} = 2$ (i.e., all the orig vertices with the coordinator) for every $1 \leq i \leq n$ and $D_{i,n+l} = [1, 3]$ (i.e., the orig with the colored vertices) for every $1 \leq l \leq k$. This forces each of the original vertices to be connected to one (or more) of the color vertices, but not to the coordinator.

Finally, we define the distance for every two original vertices $1 \leq i < j \leq n$, as follows

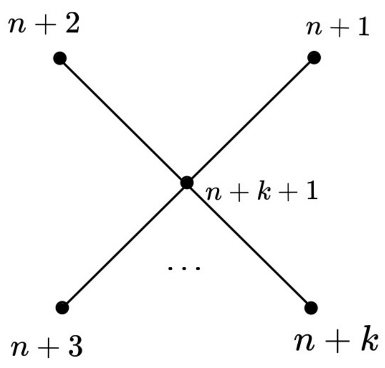


Figure 3.3: An example of the realization graph satisfying the distance constraints, i.e. $D_{n+l,n+k+1} = 1$ and $D_{n+l,n+r} = 2$.

$$D_{i,j} = \begin{cases} \{4\}, & (v_i, v_j) \in E(G) \\ \{2, 4\}, & (v_i, v_j) \notin E(G) \end{cases}$$

Now it suffices to prove that the input G is k-colorable if and only if the distance matrix D is realizable.

The first direction is when G is k-colorable. We define the colors as $1, \dots, k$ and let $x : V(G) \mapsto \{1, \dots, k\}$ be the coloring function. Now for the matrix D defined from G, construct a realizing graph G' as follows. Start with a star rooted at $n+k+1$ with the k color vertices $n+1, \dots, n+k$ as leaves. Connect each original vertex i to the color vertex $x(i)$. It is easy to verify that G' is a realization for the matrix D.

In the other direction, we assume that D has a realization G'. It is important to note that the constraints of D force the color vertices to form a star rooted at $n+k+1$. Note also that each of the original vertices must be connected to at least one of the leaves of the star. We will define a coloring function for G' as follows. For every original vertex i, let l be some color vertex connected to i, and let $x(i) = l$. Also the distance constraints defined for the original vertices specify that if two original vertices i and j are connected by an edge in G, then their distance must be 4. This makes us sure that none of the color vertices are connected to both i and j, because this would make their distance 2, so i and j would not be connected in G. It is easy to verify that if $(i, j) \in E$ then i and j get different colors.

□

3.2 Star Realizations

In this section we study Set-DR in stars. We show that there exists a polynomial time algorithm that solves 2-Set-DR in stars and we present a polynomial time algorithm that solves k-Range-DR on weighted stars, for any k. We also show that 6-Set-DR on weighted stars is NP-hard.[11]

We note that 2-Set-DR can be solved efficiently.

Theorem 3.3. *[11] There exists a polynomial time algorithm for 2-Set-DR on stars.*

Proof. Assume that i is the center of the star. It follows that the weight of any edge (i, j), for every $i \neq j$ can be either $d^0_{i,j}$ or $d^1_{i,j}$. Now we define a Boolean variable x_j, where x_j=False if the weight of the edge (i, j) is $d^0_{i,j}$ and x_j=True if the weight is $d^1_{i,j}$. The rest of the entries of D are used to create a 2-SAT that is efficient if and only if there exists a star realization of D in which i is the center.

Consider two vertices $j, k \neq i$. Since there are two possible weights for the edges (i, j) and (i, k) it is clear that there are 4 possible distances from j to k.

- $d^0_{i,j} + d^0_{i,k}$
- $d^0_{i,j} + d^1_{i,k}$
- $d^1_{i,j} + d^0_{i,k}$
- $d^1_{i,j} + d^1_{i,k}$

For each one of the above 4 options, check whether it equals $d^0_{j,k}$ or $d^1_{j,k}$. This induces a truth table on the variable x_j and x_k that can be represented by at most two 2-SAT. Doing this for all pairs of vertices creates a 2-SAT that contains at most $O(n^2)$ clauses by merging all the above mentioned clauses.

Suppose that there exists a star realization P of D with i as a center. This induces an assignment to the variables Boolean that satisfies the 2-SAT. On the other side, assume that the 2-SAT that is obtained by assuming that i is the center is satisfiable. A satisfying assignment induces a star, which complies with the profile.

Of course, since there are n candidates for the center vertex, we need to run the above process n times. It is easy to verify that the total running time is $O(n^3)$.

□

We note here that the running time for unweighted case can be improved to $O(n^2)$.[11]

Theorem 3.4. *[11] There exists a polynomial time algorithm for k-Set-DR on unweighted stars, for any k.*

Proof. It is obvious that all distances in an unweighted star are either 1 or 2, one may assume that $D_{i,j} \subseteq \{1, 2\}$, for every $i \neq j$.

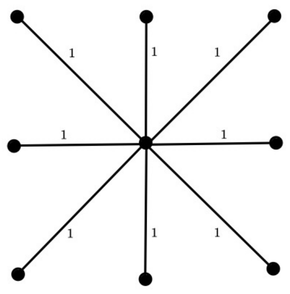


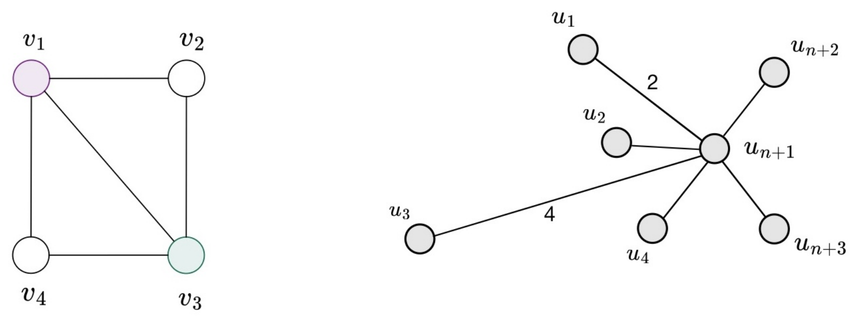
Figure 3.4

The theorem follows due to the Theorem 3.3. □

Now we show that the star realization problem is NP-hard for 6-Set-DR.[11]

Theorem 3.5. *[11] 6-Set-DR is NP-hard on weighted stars.*

Proof. To be able to prove thε theorem, we will use a reduction from the 3-coloring problem.



A 3-colorable graph A realization of D

Figure 3.5: An example of the reduction in the proof of Theorem for $n = 4$. White, purple and green corresponds to the weights of the edges of the star. Edges without a label are of weight 1.

First of all, we will consider a graph G, where $V(G) = \{v_1, \ldots, v_n\}$. We construct a distance matrix D on $n + 3$ vertices, denoted by $\{u_1, \ldots, u_{n+3}\}$. Informally, the distance matrix is defined to force the vertex u_{n+1} to be the center of the star, while the other two vertices u_{n+2} and u_{n+3} must be two of the leaves whose distance from the center is exactly 1. The rest of the vertices, $u_1, \ldots, u_n$ that are associated with the n vertices $v_1, \ldots, v_n$ of G, are leaves whose distance from the center is either 1,2 or 4. The idea is that distance 2^c is associated with the color c (i.e., 0,1 or 2). The distances between these n leaves is defined so as to assure us that the two endpoints of any edge of G are associated with different colors, if a realization exists. More specifically, the 6-set distance matrix D is defined as follows for any two indices $1 \leq i < j \leq n+3$.

$$D_{i,j} = \begin{cases} \{3,5,6\}, & i,j \leq n,\ (v_i, v_j) \in E(G) \\ \{2,3,4,5,6,8\}, & i,j \leq n,\ (v_i, v_j) \notin E(G) \\ \{1,2,4\}, & i \leq n,\ j = n+1 \\ \{2,3,5\}, & i \leq n,\ j = n+2, n+3 \\ \{1\}, & i = n+1,\ j = n+2, n+3 \\ \{2\}, & i = n+2,\ j = n+3 \end{cases}$$

We will show that G is 3-colorable if and only if D is realizable by weighted star.

We first suppose that G is 3-colorable and let $x : V(G) \mapsto \{0,1,2\}$ is the coloring function of G. We will describe a star realization G'. First of all, let u_{n+1} be the center of the star. The two vertices u_{n+2} and u_{n+3} are leaves such that $w(u_{n+1}, u_{n+2}) = w(u_{n+1}, u_{n+3}) = 1$. Next, for every $i \in \{1, \dots, n\}$, if $x(v_i) = c$, then $w(u_{n+1}, u_i) = 2^{x(v_i)}$. It is easy to verify that G' realizes D. Moreover, we observe that the first requirement of D is satisfied, since x is a 3-coloring.

In the opposite direction, we consider that G' is a star realization of D. We notice that the above distance matrix makes sure that $u_{n+2}, u_{n+1}, u_{n+3}$ form a path with two edges of weight 1. So, u_{n+1} must be the center of the star. We define a coloring x of $V(G)$ according to the weights of the edges to the center: $x(v_i) = log_2 w(u_{n+1}, u_i)$. Hence, x is a proper 3-coloring, since the firste requirement of D ensures that $W(u_{n+1}, u_i) \neq w(u_{n+1}, u_j)$ if $(v_i, v_j) \in E$. This is true because $2 = 1 + 1$, $4 = 2 + 2$ and $8 = 4 + 4$ are not members of $3, 5, 6$ which are the possible distances between u_i and u_j. (See Figure 3.5) □

In [12] it was shown that there exists a polynomial time algorithm that solves the Range-DR on a given fixed weighted tree, assuming that non-integral edges weights are allowed.

3.3 Path Realizations

In this section we study the realization of distance profiles by paths. We first show that Range-DR is NP-hard in weighted and unweighted paths and also show that Set-DR on paths is NP-hard, even on 5-set distance profiles.[11]

Before looking at the first theorem, it would be good to give a basic definition that is used. In the mathematical field of graph theory, the term

Hamiltonian path means that in a graph there is a path that passes through each vertex exactly once. More formally the definition is given below.

Definition 3.2. *A Hamiltonian path or traceable path is a path that visits each vertex of the graph exactly once. A graph that contains a Hamiltonian path is called a traceable graph. A graph is Hamiltonian-connected if for every pair of vertices there is a Hamiltonian path between the two vertices.*

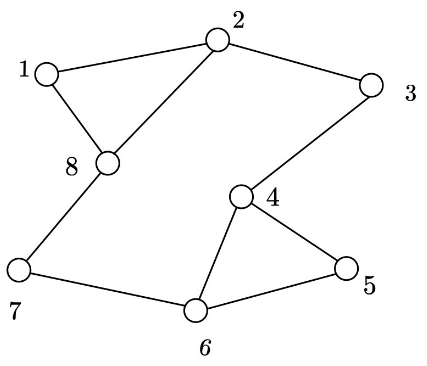


Figure 3.6: An example of a traceable graph, as it contains a Hamiltonian path, i.e. for instance the path $1 \to 2 \to 8 \to 7 \to 6 \to 5 \to 4 \to 3$.

Range-DR in weighted paths was shown to be NP-hard in [12] using a reduction from the Linear Arrangement problem. However, this result also applies to unweighted paths. It is important to note that the reduction constructs a matrix with unlimited ranges.[11]

Theorem 3.6. *[11] Range-DR is NP-hard in both weighted and unweighted paths*

Proof. We will prove the theorem using a reduction from Hamiltonian Path. Given a graph G, construct the following distance matrix

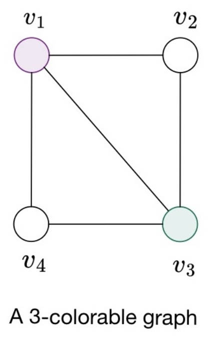
$$D_{i,j} = \begin{cases} \{1, \ldots, n-1\}, & (v_i, v_j) \in E(G) \\ \{2, \ldots, n-1\}, & (v_i, v_j) \notin E(G) \end{cases}$$

So, if G has a Hamiltonian path, then this path induces a realization of D. On the other hand, a realization of D corresponds to a Hamiltonian path in G. □

Next it turns out that the 5-Set-DR is NP-hard on paths.

Theorem 3.7. *[11] 5-Set-DR is NP-hard in unweighted and weighted paths*

Proof. To be able to prove thε theorem, we will use a reduction from the 3-coloring problem.



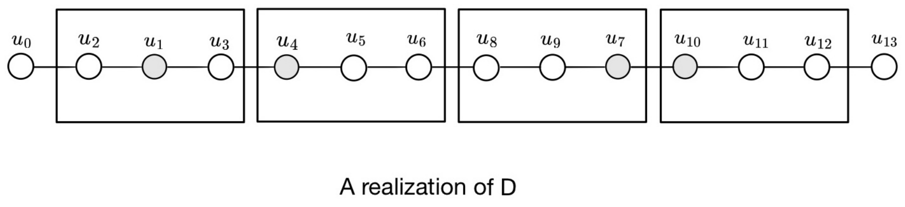
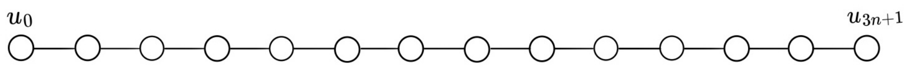


Figure 3.7: An example of the reduction in the proof of Theorem 3.5. White, purple and green are colors 0,1 and 2, respectively. The location of full nodes correspond of the full nodes correspond to the chosen colors.

First of all, we will consider a G where $V(G) = \{v_1, \dots, v_n\}$. We construct a matrix D on $3n + 2$ vertices , which we will denoted $\{u_0, \dots, u_{3n+1}\}$. We consider that the vertices $u_{3i-2}, u_{3i-1}, u_{3i}$ represent the vertex v_i in the original graph and also the location of u_{3i-2} encoded the color of v_i. Moreover the vertices u_0 and u_{3n+1} are the end-points of the path.



More formally, for every x, we define $\overline{x} \triangleq \lceil x/3 \rceil$. The matrix is defined as follows for any two vertices $0 \le i < j \le 3n+1$.

$$D_{i,j} = \begin{cases} \{3n+1\}, & i = 0,\ j = 3n+1 \\ \{3\overline{j}-2, 3\overline{j}-1, 3\overline{j}\}, & i = 0,\ \overline{j} \in \{1,\dots,n\} \\ \{3n-3\overline{i}+1, 3n-3\overline{i}+2, 3n-3\overline{i}+3\}, & \overline{i} \in \{1,\dots,n\},\ j = 3n+1 \\ \{1,2\}, & \overline{i} = \overline{j} \\ \{3(\overline{j}-\overline{i})+\Delta : \Delta \in \{-2,-1,0,1,2\}\}, & \overline{i} < \overline{j},\ i \mathbin{mod} 3 \neq 1 \\ & or\ j \mathbin{mod} 3 \neq 1 \\ & or\ (v_{\overline{i}}, v_{\overline{j}}) \notin E(G) \\ \{3(\overline{j}-\overline{i})+\Delta : \Delta \in \{-2,-1,1,2\}\}, & \overline{i} < \overline{j},\ i, j \mathbin{mod} 3 = 1, \\ & (v_{\overline{i}}, v_{\overline{j}}) \in E(G) \end{cases}$$

We will show that the G is 3-colorable if and only if D is realizable using an unweighted path.

Firstly, assume that G is 3-colorable and let $x : V(G) \mapsto \{1,\dots,3\}$ be the coloring function of G. We will consider a path realization as a placement of the vertices on integral points from 0 to $3n+1$. Intuitively, u_0 is placed on 0 and u_{3n+1} is placed on $3n+1$. Now, for every $i \in \{1,\dots,n\}$, if $x(v_i) = c$ then u_{3i-2} is placed at location $3i-2+c$. So, the vertices u_{3i-1} and u_{3i} are placed at the two remaining free locations from $\{3i-2, 3i-1, 3i\}$. It is easy to verify that P realizes D.

Now in the opposite direction, we consider that P is a path realization of D. First of all, we must be sure that the distance between u_0 and u_{3n+1} is exactly $3n+1$. Obviously, all the other distances are strictly less than $3n+1$. So, if a path realization exists, then we may assume without loss of generality that u_0 is placed on 0 and u_{3n+1} is placed on $3n+1$. In the weighted case, it follows that all other vertices are located at $\{1,\dots,3n\}$, which means that all edges are of unit length. Since the distances between u_{3i-2}, u_{3i-1} and u_{3i} are 1 or 2, for every $i \in \{1, \ldots, n\}$, these three nodes are forced to appear as a sub-path of P consisting of two edges. Moreover, the required distances from u_0 and u_{3n+1} forces u_{3i-2}, u_{3i-1} and u_{3i} to be assigned to the three consecutive positions 3i-2,3i-1,3i on the path. We will define a coloring x according to the positions of $\{u_{3i-2} : i \in \{1, \ldots, n\}\}$, i.e., $x(v_i) = c$ if u_{3i-2} is located at $3i - 2 + c$. Then x is a 3 coloring, since the last requirement of D ensures that $x(v_i) \neq x(v_j)$ if $(v_i, v_j) \in E$.

□

3.4 Cycle Realizations

Theorem 3.8. *[11] There exists a polynomial time algorithm for 2-Set-DR on weighted and unweighted cycles.*

We start the section with a hardness proof that requires unlimited ranges.

But first we must note that the term Hamilton cycle means that in the graph there is a cycle that visits each vertex once. More formally we will give the definition as follows.

Definition 3.3. *A Hamiltonian cycle, Hamiltonian circuit, vertex tour or graph cycle is a cycle that visits each vertex exactly once. A graph that contains a Hamiltonian cycle is called a Hamiltonian graph.*

Theorem 3.9. *[11] Range-DR is NP-hard in both weighted and unweighted cycles*

Proof. We will prove the theorem using a reduction from Hamiltonian Cycle. Given a graph G, construct the following distance matrix.

$$D_{i,j} = \begin{cases} \{1, \ldots, \lfloor n/2 \rfloor\}, & (v_i, v_j) \in E(G) \\ \{2, \ldots, \lfloor n/2 \rfloor\}, & (v_i, v_j) \notin E(G) \end{cases}$$

If G has a Hamiltonian cycle, then this cycle induces a realization of D. On the other side, a realization of D corresponds to a Hamiltonian cycle in G.

□

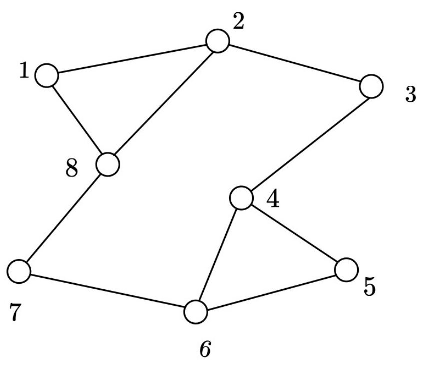


Figure 3.8: An example of a Hamiltonian graph, as it contains a Hamiltonian cycle, i.e. for instance the cycle $2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 1 \to 2$.

Now we show that 5-Set-DR on cycles is NP-hard, using the reduction from the problem on paths, where the required end points of the paths are given in the input.[11]

Theorem 3.10. *[11] 5-Set-DR is NP-hard in unweighted and weighted cycles*

Proof. To be able to prove the theorem we will use a reduction from the Set-DR in unweighted paths.

Let D be a 5-set distance matrix and the required end-points of the path are given in the input. We will add $3n$ vertices $n+1, \ldots, 4n$, unit weight edges between i and $i+1$, for $i \in \{n, \ldots, 4n-1\}$ and the unit weight edge $(4n, 1)$. Now, given a matrix D with dimensions $n \times n$, we construct a matrix D' with dimensions $4n \times 4n$, as follows.

$$D'_{i,j} = \begin{cases} D_{i,j}, & 1 \le i, j \le n \\ min\{(j-i), (4n-j+i)\}, & n < i < j \le 4n \\ \{min\{(j-\ \delta-1), (4n+1-j+\ \delta)\} :\ \delta \in D_{i,1}, & 1 \le i \le n, \\ & \quad n < j \le 4n \end{cases}$$

We assume without loss of generality that $\delta \in D_{i,1}$ if and only if $n-1-\ \delta \in D_{i,n}$.

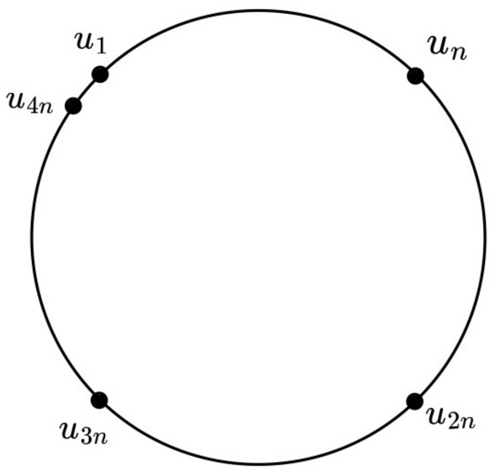


Figure 3.9: Realization of the reduction from the path realization to cycle. The line between u_1 and u_n corresponds to the location of the original vertices.

The first direction is that D is realizable. Then, D' is realizable by using a cycle of total length $4n$, where the vertices $u_1, \dots, u_n$ are placed at positions $1, \dots, n$ as they are placed in the path realization. And each vertex u_i, for every $i > n$ is placed at location i.

The other direction is that D' is realizable, and assume that u_1 and u_n are placed at locations 1 and n on the cycle. It follows that each vertex u_i, for every $i > n$ is located at i. In this case, D can be realized by the arc from 1 to n. □

CHAPTER 4

Maximum Degree in Vertex Neighborhoods

In many application domains involving networks, it is common to view vertex degrees as a central parameter, providing useful information concerning the relative significance of each vertex with respect to the rest of the network. Given an n-vertex graph G with adjacency $Adj(G)$, its degree sequence is a sequence consisting of its vertex degrees, $\text{Deg}(G) = (d_1, \ldots, d_n)$. So, given a graph G or its adjacency matrix, it is easy to extract the degree sequence. A sequence for which there exists a realization is called a graphic sequence[10]. Havel and Hakimi [23, 26] designed an efficient algorithm that given a sequence of integers computes a realizing graph, if such a graph exists.

4.1 Realizing maximum closed neighborhood degree profiles

In this section we provide a characterization of MaxNDeg profiles. For example, we discuss the uniform scenario of $\sigma = (d^k)$. We can easily observe that a star graph $K_{1,d}$ is MaxNDeg realization of the profile (d^{d+1}). By the term $K_{1,d}$ we mean the bipartite graph defined as follows.

Definition 4.1. *In the mathematical field of graph theory, a bipartite graph (or bigraph) is a graph whose vertices can be divided into two disjoint and independent sets U and V, that is, every edge connects a vertex in U to one in V. Vertex sets U and V are usually called the parts of the graph. Equivalently, a bipartite graph is a graph that does not contain any odd-length cycles.*

Now, we will show in the following lemma that, by identifying together vertices in different copies of $K_{1,d}$, it is possible to realize the profile (d^k), whenever $k \geq d+1$.

Lemma 4.1. *[10] For any positive integers $d, k > 1$, the profile $\sigma = (d^k)$ is* MaxNDeg *realizable whenever $k \geq d+1$. Also, we can always compute in $O(k)$ time a connected realization that has an independent set, let S, of size d such that all vertices in S have degree at most 2, and at least two vertices in S have degree 1.*

Proof. Let a be the smallest integer such that $k \leq 2 + a(d-1)$. We construct a caterpillar (A caterpillar is a tree in which all the vertices are within distance one of a central path) T as follows. Firstly, take a path $P = (s_0, s_1, \ldots, s_a, s_{a+1})$ of length $a+1$. Connect each internal vertex $s_i, i \in [1,a]$, with a set of $d-2$ new vertices, so that the degree of s_i is d. It is important to note that the MaxNDeg of each vertex $v \in T$ is d.

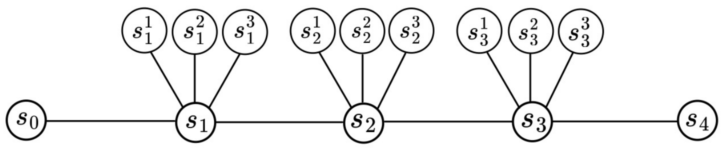


Figure 4.1: A caterpillar for $d = 5$ and $a = 3$. If $k = 12$, then $r = 2$ and we merge $(i) s_1^1, s_2^1$ and $(ii) s_1^2, s_2^2$.

If $k = 2 + a(d-1)$, then T is our required realizing graph. Now, if $k < 2 + a(d-1)$, then $a \geq 2$ since $k \geq d+1$. The "problem" now is that T has too many vertices. Let $r = 2 + a(d-1) - k$ denote the number of excess vertices in T that need to be removed. Observe that $r \leq d-2$ as otherwise we could decrease a by 1. Now, the r vertices can be removed, if we take any two distinct internal vertices s_i and s_j on P and let $(s_i^1, \ldots, s_i^{d-2})$ and $(s_j^1, \ldots, s_j^{d-2})$, respectively denote the neighbors of s_i and s_j not lying on P. Now let G be the graph obtained by merging vertices s_i^l and s_j^l into a single vertex for $l \in [1, r]$. Since the number of vertices was decreased by r, then G contains exactly n vertices. The degree of vertices $s_1, s_2, \ldots, s_a$ remains d and the degree of all other vertices is at most 2, therefore $\text{MaxNDeg}(v) = d$ for every $v \in G$. So G is a realizing graph of the profile σ.

Finally, in the graph G, the end points of P, i.e. s_0 and s_{a+1}, have degree 1 and there are $d-2$ other vertices, let $(s_i^1, \ldots, s_i^{d-2})$, that have degree bounded by 2. So we set S to these d vertices. It is easy to observe that S is an independent set and indeed the total time to compute it, is $O(k)$. □

4.1.1 An incremental procedure for computing MaxNDeg realizations

Algorithm 1: AddLayer (H, L, k, d).

```
1  Let the list $L$ be $(a_1, a_2, \ldots, a_{d-1})$.
2  Add to $H$ a set $W = w_1, \ldots, w_k$ of $k$ new vertices.
3  case $(k < d)$ do
4      Set $count = k$ and $i = d - 1$.
5      while $(count > 0)$ do
6          Let $r = min\{d - deg(a_i), count\}$.
7          Add edges $(a_i, w_{count-t})$ to $H$ for $t \in [0, r-1]$.
8          Decrement $i$ by 1 and $count$ by $r$.
9      end
10     foreach $j \in [d-1, \ldots, 2, 1]$ do
11         If $deg(a_i) = d$ then break them for loop.
12         If $(j < i)$ then add edge $(a_j, a - i)$ to $H$.
13         If $(j > i)$ then add an edge between $a_i$ and an arbitrary vertex
             in $N(a_j) \cap W$.
14     end
15     Set $L$ to be prefix of $(w_1, w_2, \ldots, w_k, a_1, a_2, \ldots, a_{i-1}$ of size $d - 2$.
16 end
17 case $(k \geq d)$ do
18     Use Lemma 4.1 to compute over independent set $(W \cup \{a_1\})$ the
         graph, say $\bar{H}$, realizing the profile $(d^{k+1})$ such that $deg_{\bar{H}}(a_1) = 1$.
19     Add edges between $a_1$ and arbitrary $d - deg(a_1)$ vertices in
         $\{a_2, \ldots, a_{d-1}\}$.
20     Let $b_1, \ldots, b_{d-1} \in \bar{H} \setminus a_1$ be such that
         $1 = deg_{\bar{H}}(b_1) \leq \cdots \leq deg_{\bar{H}}(b_{d-1}) \leq 2$.
21     Set $L = (b_1, b_2, \ldots, b_{d-2})$.
22 end
```

Below we analyze the procedure AddLayer, that is very useful in building graph realizations in a decreasing order of maximum degrees. Given a connected graph H and integers d and k satisfying $d \leq 2$ and $k \leq 1$, the procedure adds to H a set W of k new vertices such that $\text{MaxNDeg}(w) = d$ for each $w \in W$. The pseudocode is given above for the procedure AddLayer.

- *Procedure* AddLayer[10]:
  The input to procedure $\text{AddLayer}(H, L, k, d)$ is a connected graph H and a list $L = (a_1, \ldots, a_{d-1})$ of vertices in H whose degree is bounded above by $d-1$. First of all, to algorithm adds to H a set of k new vertices $W = w_1, \ldots, w_k$. So, the new vertices are connected to the vertices of L and to themselves so as to ensure that $\text{MaxNDeg}(w) = d$ for every $w \in W$. Now there are two different cases.

  Firstly, we take the case $k < d$. In this case we add edges from vertices in W to a subset of vertices from L such that those vertices in L will have degree d and so will imply $\text{MaxNDeg}(w) = d$, $\forall w \in W$. We consider two variables, *count* and i, respectively to k and $d-1$. The variable *count* holds the number of vertices in W that still need to be connected to vertices in L. While $count > 0$ the procedure does the following steps. Firstly, compute $r = min\{d - deg(a_i), count\}$, the maximum number of vertices in W that can be connected to vertex a_i. Secondly, connect a_i to following r vertices in W, i.e. $w_{count-(r-1)}, w_{count-(r-2)}, \ldots, w_{count-1}, w_{count}$. And finally, decrease *count* by r and i by 1.

  Now, when $count = 0$ the vertices $a_i, a_{i+1}, \ldots, a_{d-1}$ are connected to at least one vertex in W. It is clear that $deg(a_{d-1}) = deg(a_{d-2}) = \cdots = deg(a_{i+1}) = d$ and $deg(a_i) \leq d$. Since the input graph H was connected, in the beginning of the execution $deg(a_i) \geq 1$ and by connecting a_i to at least one vertex in W, more specifically to w_1, its degree is increased at least by one. Therefore, at most $d-2$ edges need to be added to a_i to ensure us that its degree is exactly d. The procedure does the following operation for each $j \in [d-1, d-2, \ldots, 2, 1]$ until $deg(a_i) = d$. If $j < i$ then add edge (a_j, a_i) to H and if $j > i$ then add an edge between a_i and an arbitrary neighbor of a_j lying in W. Since $deg(a_i) = deg(a_{i+1}) = \cdots = deg(a_{d-1}) = d$ and $deg(w) \leq 2, \forall w \in W$, it follows that $\text{MaxNDeg}(w) = d, \forall w \in W$. Finally, we set a new list L containing the first $d-2$ vertices in the sequence $(w_1, w_2, \ldots, w_k, a_1, a_2, \ldots, a_{i-1})$. This can become since $k + i - 1 \geq d - 2$ because $d - i \leq k$.

  Now we take the case $k \geq d$. The procedure uses Lemma 4.1 to compute over the independent set $W \cup \{a_1\}$ a graph realizing the profile (d^{k+1}) (denoted $\bar{H}$), such that $deg_{\bar{H}}(a_1) = 1$. We note that in the beginning of the execution, $deg(a_1) \in [1, d-1]$ and it is increased by one by adding $\bar{H}$ over the set $W \cup \{a_1\}$. Therefore $deg(a_1) \in [2, d]$. To ensure $deg(a_1) = d$, at most $d-2$ more edges need to be added to a_1. Edges are added between a_1 and any $d - deg(a_1)$ vertices in set $\{a_2, a_3, \dots, a_{d-1}\}$. This makes us sure that every $w \in W$ has MaxNDeg$(w) = d$. By Lemma 4.1, $\bar{H} \setminus \{a_1\}$ contains an independent set of $d-1$ vertices, say $b_1, ..., b_{d-1}$ such that $1 = deg_{\bar{H}}(b_1) \leq \cdots \leq deg_{\bar{H}}(b_{d-1}) \leq 2$. In the end, the procedure creates a new list $L = (b_1, b_2, \dots, b_{d-2})$.

It is obvious that the running time of procedure AddLayer is $O(k+d)$.[10]

For greater understanding and convenience, we denote by $H_{old}, L_{old}, H_{new}$ and L_{new} respectively the graph and the list before and after the execution of procedure AddLayer.

The next two lemmas result from the description of the algorithm.

Lemma 4.2. *[10] Each $w \in W$ satisfies* MaxNDeg$(w) = d$ *and* $N(w) \subseteq W \cup L_{old}$.

Lemma 4.3. *[10] Each $a \in L_{old} \setminus L_{new}$ satisfies $deg_{H_{new}}(a) \leq d$ and each $a \in L_{old} \cap L_{new}$ satisfies $deg_{H_{new}}(a) \leq deg_{H_{old}}(a) + 1$.*

The inheritance property We have shown above that given an independent list of $d-1$ vertices of degree at most $d-1$ in a graph H, we can add $k \geq 1$ vertices to H such that the MaxNDeg of these k vertices is d. In order to use this algorithm to add vertices of smaller MaxNDeg values we require that the list L_{new}, computed by procedure AddLayer, should satisfy the next three constraints. First, the size of L_{new} should be $d-2$. Second, the vertices of L_{new} should form an independent set and third, the vertices in L_{new} should have degree at most $d-2$.

In order to make sure these constraints on L_{new}, we impose the constraint that the list L_{old} is a valid list. Given formally the definition.

Definition 4.2. *A list $L = (a_1, a_2, \dots, a_t)$ in a graph G is said to be valid with respect to G if the following two conditions hold.*

1. *For each $i \in [1, t], deg(a_i) \leq i$.*
2. *The vertices of L form an independent set in G.*

Lemma 4.4. *[10] If the input list L_{old} in procedure* AddLayer *is valid, then the output list L_{new} is valid as well.*

Proof. First of all, we take the case $k \leq d-1$. Let i be the smallest index such that vertices $a_i, a_{i+1}, \ldots, a_{d-1}$ are adjacent to some vertex of W in H_{new}. Note that in the graph H_{new}, $w_1 \in W$ is a neighbor of a_i. Moreover, to increase the degree of a_i to d, we connect a_i to some or all vertices in $a_1, \ldots, a_{i-1}$ and some or all neighbors of $a_{i+1}, \ldots, a_{d-1}$ lying in W. So the vertex set $W \cup \{a_1, \ldots, a_{i-1}\}$ is independent in H_{new}. Also its size is at least $d-1$, since we showed that $k \geq d-i$. Since the list $L_{old} = (a_1, a_2, \ldots, a_{d-1})$ is valid in the beginning of the execution of procedure AddLayer, it results that in $H_{old}, deg(a_j) \leq j, \forall j \in [1, d-1]$. So by Lemma 4.3, in H_{new} we have $deg(a_j) \leq j+1, \forall j \in [1, i-1]$. Also by construction we have $deg(w_1) = 1$ and the degree of every vertex in $W \setminus w_1$ is at most 2. Since $deg(a_j) \leq j+1, \forall j \in [1, i-1]$, the list $(w_1, \ldots, w_k, a_1, \ldots, a_{i-1})$ is valid and has length at least $d-1$. Truncating this to length $d-2$ again gives us a valid list.

We now take the case $k \geq d$. Note again that we use Lemma 4.1 to compute over the independent set $W \cup \{a_1\}$ a graph realizing the profile (d^{k+1}). Graph $H[W \cup \{a_1\}]$ contains an independent set $\{b_1, \ldots, b_{d-1}\} \subseteq W$ such that $deg(b_1) = 1$ and for $j \in [2, d-1], deg(b_j) \leq 2$. So, $(b_1, \ldots, b_{d-2})$ is the required valid list in H_{new}. □

In the next sentence, we summarize what we have said so far.

Proposition 4.1. *For every integers $d \geq 2, k \geq 1$, and every connected graph H containing a valid list L of size $d-1$, procedure* AddLayer *adds to H in $O(k+d)$ time, a set W of k new vertices such that* MaxNDeg$(w) = d, \forall w \in W$*. All the edges added to H lie in $L \times (W \cup L)$. Also $deg_H(a) \leq d, \forall a \in L$ and the updated graph remains connected and contains a new valid list of size $d-2$.*

4.1.2 The main algorithm

We will present the main algorithm for computing the realizing graph using procedure AddLayer.[10]

Let $\sigma = (d_l^{n_l}, \ldots, d_1^{n_1})$ be any profile satisfying $d_l \leq n_l - 1$ and $d_1 \geq 2$. The construction of a connected graph realizing σ is as follows. We need the Lemma 4.1 to set G to be the graph realizing the profile $(d_l^{n_l}$. Note that G

contains an independent set, let $W = \{w_1, w_2, \ldots, w_{d_l}\}$, which satisfies the condition that the degree of the first two vertices is 1. Also the degree of the other vertices is at most 2. Now, holds that $L_{l-1} = (w_1, w_2, \ldots, w_{d_{l-1}-1})$ and recall that $d_{l-1} - 1 \leq d_l$. It is obvious that this list is valid.

Now, we note that $\forall i = l-1$ to 1, make the following steps.

1. Execute procedure AddLayer (G, L_i, n_i, d_i) on the valid list L_i of size $d_i - 1$ to add a set of n_i new vertices to G. The procedure returns a valid list L_{i-1} if size $d_i - 2$.

2. Truncate the list L_{i-1} to contain only the first $d_{i-1} - 1$ vertices. The list remains valid since any prefix of a valid list is always valid.

Algorithm 2: MaxNDeg realization of $\sigma = (d_l^{n_l}, \ldots, d_1^{n_1})$.

```
input  : A sequence $\sigma = (d_l^{n_l}, \ldots, d_1^{n_1})$ satisfying $d_l \leq n_l - 1$ and
         $d_1 \geq 2$.
1 Initialize $G$ to be the graph obtained from Lemma 4.1 that realizes the
    profile $(d_l^{n_l})$.
2 Let $L_{l-1}$ be a valid list in $G$ of size $d_{l-1} - 1$.
3 for ($i = l - 1$ to 1) do
4     $L_{i-1} \to$ AddLayer($G, L_i, n_i, d_i$).
5     Truncate list $L_{i-1}$ to contain only the first $d_{i-1} - 1$ vertices.
6 end
  output: $G$.
```

Now, we will prove the correctness.[10]

Let V_l be the set of vertices in graph initialized i step 1. For every $i \in [1, l-1]$ let V_i be the set of n_i new vertices added to the graph in iteration i of the for loop. Moreover $\forall i \in [1, l]$ let G_i denote the graph induced by vertices $V_i \cup \cdots \cup V_l$. The lemma below finally proves the correctness.

Lemma 4.5. *[10] For every $i \in [1, l], G_i$ is a* MaxNDeg *realization of profile $(d_l^{n_l}, \ldots, d_i^{n_i})$ and for every $j \in [i, l]$ and $v \in V_j, deg_{G_i}(v) \leq$* MaxNDeg$_{G_i}(v) = d_j$.

Proof. In order to prove the claim, we use the inductions on the iterations of the for loop. The main case is for index l, and using Lemma 4.1, we have that $deg_{G_l}(v) \leq$ MaxNDeg$_{G_l}(v) = d_l, \forall v \in V_l$. For the step which is inductive, we consider that the claim holds for $i+1$ and we prove the claim for i. Let any vertex v in G_i. So, we have two different cases.

1. $\forall v \in V_i$ In this case we have that $deg_{G_i}(v) \leq \textsc{MaxNDeg}_{G_i}(v) = d_i$ by using Proposition 4.1.

2. $\forall v \in V_j$, *where* $j > i$ For every vertex $w \in N_{G_i}[v], deg_{G_i}(w) \leq d_j$. If $w \in V_i$, then we have already proved that $deg_{G_i}(w) \leq d_i$. Therefore, we assume the case $w \in V_{i+1} \cup \cdots \cup V_l$. So, if $w \in L_i$ participates in procedure AddLayer (G, L_i, n_i, d_i), and by Proposition 4.1, in the updated graph $deg_{G_i}(w) \leq d_i \leq d_j$. Now, if $w \notin L_i$, the degree of w is fixed in the i^{th} iteration and $deg_{G_i}(w) = deg_{G_{i+1}}(w) \leq \textsc{MaxNDeg}_{G_{i+1}}(v) = d_j$. So, MaxNDeg$(v)$ remains fixed thanks to iteration i and $\textsc{MaxNDeg}_{G_i}(v) = \textsc{MaxNDeg}_{G_{i+1}}(v) = d_j$.

Since, for every $j \in [i,l], \textsc{MaxNDeg}_{G_i}(v) = d_j$, we take that G_i is a MaxNDeg realization of profile $(d_l^{n_l}, \ldots, d_i^{n_i})$. □

The running time of the algorithm is $O(\sum_{i=1}^{l}(n_i+d_i))$ and this is optimal.[10]

Indeed, every connected graph realizing σ must contain $\Omega(n_1+n_2+\cdots+n_l)$ edges, since the degrees of all vertices must not be zero. Moreover, the graph must contain at least one vertex of each of the degrees $d_1, d_2, \ldots, d_l$ and must also have $\Omega(d_1 + d_2 + \cdots + d_l)$ edges. Any realizing graph must contain $\Omega(\sum_{i=1}^{l}(n_i + d_i))$ edges and the running time must be at least $\Omega(\sum_{i=1}^{l}(n_i + d_i))$.

From what we have mentioned so far the following theorem follows.

Theorem 4.1. *[10] There exists an algorithm where;*
Given any profile $\sigma = (d_l^{n_l}, \ldots, d_1^{n_1})$ satisfying $d_l \leq n_l - 1$ and $d_1 \geq 2$ computes in optimal time a connected MaxNDeg *realization of σ.*

4.1.3 A full characterization for MaxNDeg realizable profiles

In this subsection we will first see the necessary conditions for MaxNDeg realizability.

Lemma 4.6. *[10] A necessary condition for a profile $\sigma = (d_l^{n_l}, \ldots, d_1^{n_1})$ to be* MaxNDeg *realizable is $d_l \leq n_l - 1$.*

Proof. We consider that σ is MaxNDeg realizable by a graph G. So, G must contain a vertex, let w, of degree d_l. Since, d_l is the maximum degree in G, the MaxNDeg of all the $d_l + 1$ vertices in $N[w]$ must be d_l. Therefore, $n_l \geq d_l + 1$. □

Now if we consider a profile $\sigma = (d_l^{n_l}, \dots, d_1^{n_1})$ realizable by a connected graph. If we take the case $d_1 = 1$, then the graph must contain a vertex, let w, of degree 1. And also the vertices in his closed neighborhood, i.e. $N[v]$, must have degree 1. It is easy to verify that the only realizable graph is a single edge graph on two vertices, i.e. $\sigma = (1^2)$. Now, for $d_1 \geq 2$, then by Lemma 4.6, for any σ to be realizable in this case we need that $n_l \geq d_l + 1$. Moreover, using the Theorem 4.1, under these two conditions σ is always realizable.

Theorem 4.2. *[10] A profile $\sigma = (d_l^{n_l}, \dots, d_1^{n_1})$ is* MaxNDeg *realizable by a connected graph if and only if one of the following holds.*

1. *$d_l \leq n_l - 1$ and $d_1 \geq 2$ or*
2. *$\sigma = (1^2)$, i.e. $d_1 = 1$.*

Note that in the case where $d_1 = 1$, then n_1 must be even, since the vertices v with MaxNDeg$(v) = 1$ must form a disjoint union of exactly $n_1/2$ edges.

Theorem 4.3. *[10] A profile $\sigma = (d_l^{n_l}, \dots, d_1^{n_1})$ is* MaxNDeg *realizable by a general graph if and only if $d_l \leq n_l - 1$ and holds one of the following.*

I. n_1 is even or

II. $d_1 \geq 2$.

4.2 Realizing maximum open neighborhood degree profiles

This problem of open neighborhoods is not too different from the previous problem of the closed neighborhoods and we can deal with it with similar techniques. However, open neighborhoods have some technical issues. Note that in the incremental algorithm we saw in the previous section, we use both degree one as well as higher degree vertices of the partially constructed graph to obtain a MaxNDeg realization. The number of degree one vertices available to work with is smaller for a MaxNDeg profile. To handle this, we modify our approach and introduce the pseudo-valid lists.[10]

4.2.1 Pseudo-valid list

In this section we will first give some lemmas which are an extension of Lemma 4.1 and Proposition 4.1 presented in the previous section for MaxNDeg profiles.

Lemma 4.7. *[10] For every positive integers d and k, the profile $\sigma = (d^k)$ is* MaxNDeg$^-$ *realizable whenever $k \geq d + 2$. Also, we can compute in $O(k)$ time a connected realization that contains an independent set having, first, two vertices of degree 1, and second, $d - 2$ other vertices of degree at most 2.*

Proposition 4.2. *For every integers $d \geq 2, k \geq 1$ and every connected graph H containing a valid list L of size $d - 1$, procedure* AddLayer *adds to H a set W of k new vertices such that* MaxNDeg$^-(w) = d, \forall w \in W$*, in $O(k + d)$ time. All the edges added to H lie in $L \times (W \cup L)$. Also, $deg_H(a) \leq d, \forall a \in L$ and the updated graph remains connected and contains a new valid list of size $d - 2$.*

It is necessary to note that the Proposition 4.2 holds for the open neighborhoods and can not be used to incrementally compute the realizations. For example, for $\sigma = (d_l^{d_l+1})$ unlike the scenario of MaxNDeg realization, there is no MaxNDeg$^-$ realization that contains a valid list. For this reason we define pseudo-valid lists.

Definition 4.3. *A list $L = (a_1, a_2, \dots, a_t)$ in a graph H is said to be pseudo-valid with respect to H if,*

1. *for each $i \in [1, t], deg(a_i) = 2$ and*
2. *the vertices of L form an independent set.*

It is easy to verify that the only deviation that prevents L from being a valid list is that $deg(a_1) = 2$ instead of 1.

Lemma 4.8. *[10] For every two integers $d > \bar{d} \geq 2$, the profile $\sigma = (d^d, \bar{d}^1)$ is* MaxNDeg$^-$ *realizable. Also, in $O(d)$ time we can compute a connected realization that contains a valid list of size $d - 1$.*

Proof. We need to construct the graph G. So, we consider a vertex z and connect it to $d - 1$ other vertices $v_1, \dots, v_{d-1}$. We consider now another vertex, say y, and connect to $v_1, \dots, v_{\bar{d}-1}$. Moreover connect z to the vertex y. In the graph G, we have that $deg(z) = d, deg(y) = \bar{d}$ and $deg(v_i) \leq 2, \forall i \in [1, d-1]$. Note that v_{d-1} is not adjacent to y, since $\bar{d} < d$, so $deg(v_{d-1}) = 1$. It is easy to verify that $\textsc{MaxNDeg}^-(z) = \bar{d}, \textsc{MaxNDeg}^-(y) = d$ and $\textsc{MaxNDeg}^-(v_i) = d, \forall i \in [1, d-1]$. Therefore, the list $(v_{d-1}, \dots, v_{\bar{d}-1}, \dots, v_2, v_1)$ is valid in G. $\square$

Lemma 4.9. *[10] For every integer $d \geq 2$, the profile $\sigma = (d^{d=1})$ is $\textsc{MaxNDeg}^-$ realizable. Also, a connected realization that contains an independent set having $d-1$ vertices of degree 2 can be computed in $O(d)$ time. However, none of the graphs realizing σ can contain a vertex of degree 1.*

Proof. We follow a similar construction for the graph G as in the previous lemma. We consider two vertex sets, let $U = \{u-1, u_2\}$ and $W = \{w_1, \dots, w_{d-1}\}$. We add to G the edge (u_1, u_2) and for every $i \in [1, d-1]$, add to G the edges (u_1, w_i) and (u_2, w_i). This procedure makes us sure that $deg(u_1) = deg(u_2) = d$ and $deg(w_i) = 2, \forall i \in [1, d-1]$. Now, the graph G contains $d+1$ vertices with $\textsc{MaxNDeg}^-$ equal to d. Moreover, W is an independent set of size $d-1$ in G and $deg(w_i) = 2, \forall w_i \in W$.

Next, we consider any $\textsc{MaxNDeg}^-$ realizing graph, say H, of σ. The graph H must contain two vertices, let x and y, of degree d, since a single vertex of degree d in H makes us sure that $\textsc{MaxNDeg}^- = d$ for at most d vertices. We note that $N[x] = N[y]$, since otherwise H will contain more than $d+1$ vertices. So all the vertices other x and y are adjacent to both x and y. Therefore, every vertex in H must have at least degree 2. $\square$

In the next lemma we show that $\textsc{AddLayer}$ outputs a valid list, even when the input list is pseudo-valid.[10]

Lemma 4.10. *[10] In procedure* $\textsc{AddLayer}$ *the list L_{new} is valid list, even when the list L_{old} is pseudo-valid and the parameter d satisfies $d \geq 3$.*

Proof. We will use the proof of Lemma 4.4. We have again two different cases depending on whether or not $k < d$. Firstly, we take the case $k \leq d-1$. It is valid from Lemma 4.4 that $(w_1, \dots, w_k, a_1, \dots, a_{i-1})$ is a valid list of length at least $d-1$ when $deg_{H_{old}}(a_1) = 1$. We take the case when L_{old} is pseudo-valid and $deg_{H_{old}}(a_1) = 2$. The list L_{new} is still valid if $k \geq 2$ since the degree of a_1 in H_{new} is at most 3 and the position in L_{new} is 3 or greater. So the case is $k = 1$. In this case $i = d-1$, since the only vertex w_1 belonging to W is connected to a_{d-1} in Algorithm 1. Moreover, $deg_{H_{old}}(a_{d-1}) = 2$ and a_{d-1} is connected to vertex w_1, so to makes us sure that $deg(d_{d-1}) = d$. Since a_{d-1} is never connected to vertex $a_1, deg_{H_{new}}(a_1) = deg_{H_{old}}(a_1) = 2$. This shows that the sequence $(w_1, \ldots, w_k, a-1, \ldots, a_{i-1}) = (w_1, a_1, \ldots, a_{d-2})$ is a valid list of length $d-1$. Truncating it to length $d-2$ results a valid sequence. In case $k \geq d$, $a_1's$ degree does not play role, therefore the argument from Lemma 4.4 works as it is. □

Remark 4.1. *The condition $d \geq 3$ is very important in Lemma 4.10 because in a pseudo-valid list all the vertices have degree exactly 2. However, procedure* AddLayer *works only in the case when the degree of each vertex in the list is at most $d-1$, which does not be true for a pseudo-valid list when $d = 2$. Therefore, we give a different method for the profile $(d^{d=1}, 2^k)$.*

4.2.2 MaxNDeg$^-$ realization of the profile $\sigma = (d^{d+1}, 2^k)$

The next lemmas shows that $\sigma = (d^{d+1}, 2^1)$ is not MaxNDeg$^-$ realizable when $d \geq 3$, but $\sigma = (d^{d+1}, 2^k)$ is MaxNDeg$^-$ realizable when $d \geq 3$ and $k \geq 2$.[10]

Lemma 4.11. *[10] For any integer $d \geq 3$, the profile $\sigma = (d^{d+1}, 2^1)$ is not* MaxNDeg$^-$ *realizable.*

Proof. We first assume that σ is MaxNDeg$^-$ realizable by a graph G. Also we consider a vertex $w \in V(G)$, such that MaxNDeg$^-(w) = 2$. The graph must contain at least 2 vertices, let x and y, of degree d, since a single vertex of degree d can guarantee MaxNDeg$^-$ of d for at most d vertices in the graph. We take the next 2 cases.

1. $N[x] = N[y]$ In this case the MaxNDeg$^-$ of all vertices in $N[x] = N[y]$ is at least $d \geq 3$, since they are adjacent to one of the vertices x or y. So, $w \notin N[x]$, which implies that $V(G) = N[x] \cup \{w\}$, since $|N[x]| = d+1$ and $|V(G)| = d+2$. Moreover, w cannot be adjacent to any vertex in $N[x]$, because if w is adjacent to a vertex $w_0 \in N[x]$, then $deg(w_0) = 3$, in contradiction to the assumption MaxNDeg$^-(w) = 2$. Therefore, the only possibility left is that w is a singleton vertex, which is a contradiction.

2. $N[x] \neq N[y]$ In this case the vertex set of G is equal to $N[x] \cup N[y]$ since his size must be at least $d+2$ and is also at most $|V(G)| = d+2$. This means that all the vertices of G are adjacent to one of the vertices x or y, which contradicts the fact that MaxNDeg$^-(w) = 2$, since $deg(x) = deg(y) = d \geq 3$.

□

Lemma 4.12. *[10] For any integers $d \geq 3$ and $k \geq 2$, the profile $(d^{d+1}, 2^k)$ is* MaxNDeg$^-$ *realizable. Also, we can compute a connected realization in $O(d+k)$ time.*

Proof. We have to construct the graph G. We consider a vertex u_1 and connect it to d other vertices $v_1, \dots, v_d$. Next, consider another vertex u_2 and connect it to vertices $v_2, \dots, v_d$, and a new vertex v_{d+1}. Finally we take a path $(a_1, a_2, \dots, a_a)$ on $a = k-2$ new vertices and connect a_1 to v_{d+1}. In the graph holds that $deg(u_1) = deg(u_2) = d$ and $deg(v_i), deg(a_j) \leq 2, \forall i \in [1, d+1]$ and $j \in [1, k-2]$. Now, the vertices u_1, u_2 have maximum degree in their neighborhood 2, so MaxNDeg$^-(u_1) =$ MaxNDeg$^-(u_2) = 2$, i.e. (2^2). Every vertex v_i is adjacent to $u_1, u_2, \forall i \in [1, d+1]$, so its MaxNDeg$^-$ is d, i.e. (d^{d+1}). And the MaxNDeg$^-$ of vertices on the path $(a_1, a_2, \dots, a_a)$ is 2, i.e. (2^{k-2}), since they have a neighbor of degree 2. □

4.2.3 The main algorithm

We give instructions for the construction of a graph realizing the profile $\sigma = (d_l^{n_l}, \dots, d_1^{n_1}) \neq (d_l^{d_l+1}, 2^1)$ that satisfies that $d_l \leq min\{n_l, n-1\}$ and $d_1 \geq 2$, where $n = n_1 + \cdots + n_l$. If σ is equal to $(d_l^{d_l+1}, 2^k)$ for any $k \geq 2$, we use Lemma 4.12 to realize σ.

If not, we realize the graph G differently as follows (see Algorithm 3 below for the pseudocode).

1. If $n_l \geq d_l + 2$, we use Lemma 4.7 to initialize G to be a MaxNDeg$^-$ realization of the profile $(d_l^{n_l})$. Note that G contains an independent set, let $W = \{w_1, \dots, w_{d_l}\}$, satisfying the case that the degree of the first 2 vertices is 1 and the degree of the remaining vertices is at most 2. We consider L_{l-1} to be the list $(w_1, \dots, w_{d_{l-1}-1})$. It is obvious that this list is valid.

2. If $n_l = d_l + 1$, then a realization of $(d_l^{d_l+1}$ does not contains a valid list. Therefore we use Lemma 4.9 to initialize G to be a MaxNDeg$^-$ realization of the profile $(d_l^{d_l+1})$ that contains a pseudo-valid list. We showed that G contains an independent set, let $W = \{w_1, w_2, \dots, w_{d_l-1}\}$, such that degree of each $w \in W$ is 2. We set L_{l-1} to be the list $(w_1, w_2, \dots, w_{d_{l-1}-1})$.

3. If $n_l = d_l$, then the sequence $(d_l^{d_l})$ is not realizable. Therefore we initialize G to be the graph realization of $(d_l^{n_l}, d_{l-1})$ as obtained from Lemma 4.8. We consider L_{l-1} to be a valid list in G of size $d_{l-1} - 1$ and we reduce n_{l-1} by 1, since G contain a vertex with $\textsc{MaxNDeg}^- = d_{l-1}$.

Now, $\forall i = \{l-1, \dots, 1\}$ we make the following steps.

1. We take as an input the valid list L_i of size $d_i - 1$ and execute procedure $\textsc{AddLayer}$ (G, L_i, n_i, d_i) to add n_i new vertices to G. The procedure returns a valid list L_{i-1} of size $d_i - 2$.

2. Truncate list L_{i-1} to contain only the first $d_{i-1} - 1$ vertices. The list now remains valid since it is a prefix of a valid list.

Correctness Let $\bar{V}_l$ be the set of vertices in graph G initialized in each step of the 3 cases for the value of n_l, i.e. in steps 6, 10 and 14, of Algorithm 3. For every $i \in [1, l-1]$, let $\bar{V}_i$ be the set of new vertices added to graph G in iteration i of for loop. For every $i \in [1, l]$, let G_i be the graph induced by vertices $\bar{V}_i \cup \dots \cup \bar{V}_l$.

We note that if $n_l = d_l$, then the graph is initialized in the first step of the case where $n_l = d_l$, i.e. in step 14, and contains $n_l + 1$ vertices , of which one vertex, let z has $\textsc{MaxNDeg}^-(z) = d_{l-1}$ and the other vertices have $\textsc{MaxNDeg}^- = d_l$. If $n_l = d_l$, then let $Z = \{z\}$, differently $Z = \emptyset$. We set $V_l = \bar{V}_l \setminus Z, V_{l-1} = \bar{V}_{l-1} \cup Z$ and $V_i = \bar{V}_i, \forall i \in [1, l-2]$. So $|V_i| = n_i, \forall i \in [1, l]$.

In [10] is given a Lemma which proves the correctness (we will omit it).

Algorithm 3: MaxNDeg$^-$ realization of $\sigma = (d_l^{n_l}, \ldots, d_1^{n_1})$.

```
input  : A sequence $\sigma = (d_l^{n_l}, \ldots, d_1^{n_1}) \neq (d_l^{d_l+1}, 2^1)$ satisfying $d_1 \geq 2$
         and $d_l \leq min\{n-1, n_l\}$.
1  if $\sigma = (d_l^{d_l+1}, 2^k)$ for some $k \geq 2$ then
2      Use Lemma 4.12 to compute a realization $G$ for profile $\sigma$.
3  end
4  else
5      case $n_l \geq d_l + 2$ do
6          Let $G$ be graph obtained from Lemma 4.7 that realizes profile
            $(d_l^{n_l})$.
7          Set $L_{l-1}$ to be a valid list in $G$ of size $d_{l-1} - 1$.
8      end
9      case $n_l = d_l + 1$ do
10         Let $G$ be graph obtained from Lemma 4.8 that realizes profile
            $(d_l^{d_l+1})$.
11         Set $L_{l-1}$ to be a pseudo-valid list in $G$ of size $d_{l-1} - 1$.
12     end
13     case $n_l = d_l$ do
14         Let $G$ be graph obtained from Lemma 4.9 that realizes profile
            $(d_l^{d_l}, d_{l-1})$.
15         Set $L_{l-1}$ to be a valid list in $G$ of size $d_{l-1} - 1$.
16         Decrement $n_{l-1}$ by 1.
17     end
18     for $(i = l - 1$ to $1)$ do
19         $L_{i-1} \rightarrow$ AddLayer $(G, L_i, n_i, d_i)$.
20         Truncate list $L_{i-1}$ to contain only the first $d_{i-1} - 1$ vertices.
21     end
22 end
output: $G$.
```

4.2.4 A full characterization for MaxNDeg$^-$ realizable profiles

At this point we will give some basic and sufficient conditions for a profile to be MaxNDeg$^-$ realizable.

Lemma 4.13. *[10] A necessary condition for the profile* $\sigma = (d_l^{n_l}, \dots, d_1^{n_1})$ *with* $n = n_1 + \dots + n_l$ *to be* MaxNDeg$^-$ *realizable is* $d_l \leq min\{n_l, n-1\}$.

Proof. Consider that σ is MaxNDeg$^-$ realizable by a graph H. So there exists at least 1 vertex, let u of degree d_l in the graph. Thus, $|N(u)| = d_l$ and $|N[u]| = d_l + 1$, which means that the number of vertices in the graph with MaxNDeg$^- = d_l$ must be at least d_l, so $n_l \geq d_l$. Moreover, the number of vertices in H, i.e. n, must be at least $d_l + 1$. □

We suppose a profile $\sigma = (d_l^{n_l}, \dots, d_1^{n_1})$ realizable by a graph G. If $d_1 = 1$ then the realizing graph must contain a vertex, let x, such that each vertex in $N(x)$ has degree 1. Let $deg(x) = d$ and $v_1, \dots, v_d$ be the neighbors of x. Then $deg(v_1) = \dots = deg(v_d) = 1$. Therefore in this case the realizing graph is a star $k_{1,d}$ with MaxNDeg$^-$ profile $\sigma = (d^d, 1^1)$. Now, if $d_1 \geq 2$ then by Lemma 4.13, for σ to be realizable, we need that $d_l \leq min\{n_l, n-1\}$. Lemma 4.11 makes us sure that σ must not be $(d^{d+1}, 2^1)$, but we know that σ is always realizable.[10]

Theorem 4.4. *[10] A profile* $\sigma = (d_l^{n_l}, \dots, d_1^{n_1}) \neq (d^{d+1}, 2^1)$ *with* $n = n_1 + \dots + n_l$ *is* MaxNDeg$^-$ *realizable by a connected graph if and only if one of the following holds.*

1. $d_l \leq min\{n_l, n-1\}$ *and* $d_1 \geq 2$ *or*
2. $\sigma = (d^d, 1^1)$ *for some positive integer* $d > 1$ *or*
3. $\sigma = (1^2)$.

For general graphs holds the next theorem.

Theorem 4.5. *[10] A profile* σ *is* MaxNDeg$^-$ *realizable by a general graph if and only if* σ *can be split into two profiles* σ_1 *and* σ_2 *such that both of the following holds.*

1. σ_1 *has a connected* MaxNDeg$^-$ *realization and*
2. $\sigma_2 = (1^{2a})$ *or* $\sigma_2 = (d^d, 1^{2a+1})$ *for some integers* $d \geq 2, a \geq 0$.

4.3 Counting the numbers of realizable sequences

We denote $\text{CC}\textsc{on}(n)$ and $\text{OC}\textsc{on}(n)$ the number of length n sequences that are $\textsc{MaxNDeg}$ and $\textsc{MaxNDeg}^-$, respectively, realizable by a connected graph. Also, we again denote as $\text{CG}\textsc{en}(n)$ and $\text{OG}\textsc{en}(n)$ the number of length n sequences that are $\textsc{MaxNDeg}$ and $\textsc{MaxNDeg}^-$, respectively, realizable by a general graph. The following theorem summarizes the results. [10]

Theorem 4.6. *[10] For $n \geq 5$ holds that,*

1. $\text{CC}\textsc{on}(n) = 2^{n-3}$
2. $\text{OC}\textsc{on}(n) = 2^{n-2} - 1$
3. $\text{CG}\textsc{en}(n) = \frac{2^{n-1}+(-1)^n}{3}$
4. $\lceil \frac{2^n-2}{3} \rceil - \lceil \frac{n-4}{2} \rceil \leq \text{OG}\textsc{en}(n) \leq 2^{n-1} - 1$

First of all, we note that there are $(n-1)^n$ sequences of length n on the integers $1, \dots, n-1$. We want to count the number of non-increasing sequences, denoted by S_n. Let $f(i,j,k)$ be the number of non-increasing sequences of length k on the integers $i, \dots, j$. So, by definition, we have that $S_n = f(1, n-1, n)$.

Proposition 4.3. $f(i,j,k) = \begin{pmatrix} k+j-i \\ k \end{pmatrix}$.

4.3.1 Connected graphs in the closed neighborhood profile

As we mentioned in Theorem 4.2, $\sigma = (d_l^{n_l}, \dots, d_1^{n_1}) \in S_n$ can be realized with a connected graph in the closed neighborhood if and only if one of the following holds.

- $n \geq 3, d_l \leq n_l - 1$ and $d_1 \geq 2$ or
- $n = 2$ *and* $\sigma = (1^2)$, i.e. $d_1 = 1$.

Lemma 4.14. *[10]* $\text{CC}\textsc{on}(2) = 1$ *and* $\text{CC}\textsc{on}(n) = 2^{n-3}, \forall n \geq 3$.

Proof. By the first part of characterization, we consider that $n \geq 3$ and let $d = d_l$. By the second part, the first $d = 1$ values in every realizable sequence must be d. The sequence of length $n-d-1$ is non increasing on the numbers $2, \dots, d$. By the definition of f, for $i = 2, j = d$ and $k = n-d-1$ and by Proposition 4.3 we have the number of such sequences.

$$f(2, d, n-d-1) = \binom{(n-d-1)+d-2}{n-d-1} = \binom{n-3}{d-2}$$

The value of d ranges from 2 to $n-1$. So, the total number is

$$\textsc{CCon}(n) = \sum_{d=2}^{n-1} \binom{n-3}{d-2}$$

Now, we set $i = d-2$ and finally we have,

$$\sum_{i=0}^{n-3} \binom{n-3}{i} = 2^{n-3}$$

□

Example 4.1. *We give an example for the 4 and 8* MaxNDeg *realizable sequences of length 5 and 6 with connected graphs.*

Length 5 : $(4^5), (3^5), (3^4, 2), (2^5)$.

Length 6 : $(5^6), (4^6), (4^5, 3), (4^5, 2), (3^6), (3^5, 2), (3^4, 2^2), (2^6)$.

4.3.2 Connected graphs in the open neighborhood profile

As we mentioned in Theorem 4.4, $\sigma = (d_l^{n_l}, \dots, d_1^{n_1}) \in S_n$ can be realized with a connected graph in the open neighborhood if and only if one of the following holds.

- $n = 2$ and $\sigma = (1^2)$ or
- $n \geq 3$ and $\sigma = ((n_1)^{n-1}, 1^1)$, i.e. a star graph, or
- $n \geq 3, d_l \leq n_l, d_1 \geq 2$ and $\sigma \neq ((n-2)^{n-1}, 2^1)$.

Lemma 4.15. *[10]* $\textsc{OCon}(2) = 1, \textsc{OCon}(3) = 2, \textsc{OCon}(4) = 4$ *and* $\textsc{OCon}(n) = 2^{n-2} - 1, \forall n \geq 5$.

In [10] there is the proof of the Lemma 4.15, but we will omit it. However, as shown in the proof, it is easy to verify that $\text{OC}\textsc{on} \approx 2 * \text{CC}\textsc{on}$. This is due to the looser constraints.

Example 4.2. *We give an example for the 7 and 15* $\text{M}\textsc{ax}\text{ND}\textsc{eg}^-$ *realizable sequences of length 5 and 6 with connected graphs.*

Length 5 : $(4^4, i), \forall i \in \{4,3,2,1\}$,
$(3^5), (3^3, 2^2), (2^5)$.

Length 6 : $(5^5, i), \forall i \in \{5,4,3,2,1\}$,
$(4^4, i, j), \forall i, j \in \{4,3,2\}$,
$(3^6), (3^5, 2), (3^4, 2^2), (3^3, 2^3), (2^6)$.

4.3.3 General graphs in the closed neighborhood profile

As we mentioned in Theorem 4.3, $\sigma = (d_l^{n_l}, \ldots, d_1^{n_1}) \in S_n$ can be realized with a general connected graph in the closed neighborhood if and only if one of the following holds.

- $n \geq 2, d_l \leq n_l - 1$ and $d_1 \geq 2$ or
- $n \geq 2, d_l \leq n_l - 1$ and n_1 is even.

Lemma 4.16. *[10] For every* $n \geq 2$, $\text{CG}\textsc{en}(n) = \dfrac{2^{n-1} + (-1)^n}{3}$.

Proof. It is obvious that there are no realizable sequences of length 1. So $\text{CG}\textsc{en}(1) = 0$. Also, the only sequence of length 2 is (1^2) and so $\text{CG}\textsc{en}(2) = 1$. We now consider that $n \geq 3$. From Theorem 4.3 it results that a profile which cannot be realized by a connected graph, then must contain an isolated edge. Therefore, holds $\text{CG}\textsc{en}(n) = \text{CC}\textsc{on}(n) + \text{CG}\textsc{en}(n-2)$. And, finally we get that

$$\text{CG}\textsc{en}(n) = \text{CC}\textsc{on}(n) + \text{CG}\textsc{en}(n-2) = \text{CG}\textsc{en}(n-2) + 2^{n-3}.$$

The above claim holds for 2 different cases $n = 1$ and $n = 2$, since if we substitute in the type we have $(2^0 + (-1)^1)/3 = 0$ and $(2^1 + (-1)^2)/3 = 1$ respectively. So, consider the case $n-2$ is correct and we will take, $\text{CG}\textsc{en}(n-2) = (2^{n-3} + (-1)^{n-2})/3$. Finally, it results that

$$\textsc{CGen}(n) = (2^{n-3} + (-1)^{n-2})/3 + 2^{n-3} = (2^{n-1} + (-1)^n)/3.$$

$\square$

Example 4.3. *We give an example for the 5 and 11* MaxNDeg *realizable sequences of length 5 and 6 with general graphs.*

Length 5 : $(4^5), (3^5), (3^4, 2), (2^5), (2^3, 1^2)$.

Length 6 : $(5^6), (4^6), (4^5, 3), (4^5, 2), (3^6), (3^5, 2), (3^4, 2^2), (3^4, 1^2), (2^6), (2^4, 1^2), (1^6)$.

4.3.4 General graphs in the open neighborhood profile

This case is more difficult to study and there is no exact value of OGen(n). The main problem is that no way has been found to avoid counting a sequence more than once. For example, assume the sequence $(3^6, 2^2, 1^1)$. It can be realized by a cycle of size 4 that is connected to a vertex of degree 1 and the MaxNDeg$^-$ sequence is $(3^3, 2^2)$ and a star of size 4 and its MaxNDeg$^-$ sequence is $(3^3, 1^1)$. The problem is that if we extract the star and count the number of realizations for the other sequences would count more than once sequences from which we can extract stars.

So, [10] describes a different strategy that controls the upper and lower bounds of OGen(n) and proves that $\lceil \frac{2^n-2}{3} \rceil - \lceil \frac{n-4}{2} \rceil \leq \textsc{OGen}(n) \leq 2^{n-1} - 1$.

Example 4.4. *We give an example for the 9 and 21* MaxNDeg$^-$ *realizable sequences of length 5 and 6 with general graphs.*

Length 5 : $(4^4, i), \forall i \in \{4, 3, 2, 1\}$,
$(3^5), (3^3, 2^2), (2^5), (2^3, 1^2), (2^2, 1^3)$.

Length 6 : $(5^5, i), \forall i \in \{5, 4, 3, 2, 1\}$,
$(4^4, i, j), \forall i, j \in \{4, 3, 2\}$,
$(3^6), (3^5, 2), (3^4, 2^2), (3^4, 1^2), (3^3, 2^3)$,
$(3^3, 2, 1^2), (3^3, 1^3), (2^6), (2^5, 1), (2^4, 1^2), (1^6)$.

CHAPTER 5

Minimum Degree in Vertex Neighborhoods

In this chapter, we study again a graph realization problem that pertains to degrees in vertex neighborhoods. More specifically, we study the minimum degree in vertex neighborhood profile, which we denote it MinNDeg.

First of all, it is very important to give the necessary and sufficient conditions for a profile σ to be MinNDeg realizable.

Proposition 5.1. *[7] A profile $\sigma = (d_l^{n_l}, \dots, d_1^{n_1})$ is* MinNDeg *realizable, when the following necessary conditions hold.*

$$d_i \leq n_1 + n_2 + \cdots + n_i - 1, \forall i \in [1, l] \tag{NC1}$$

$$d_l \leq \lfloor \frac{n_1 d_1}{d_1 + 1} \rfloor + \lfloor \frac{n_2 d_2}{d_2 + 1} \rfloor + \cdots + \lfloor \frac{n_l d_l}{d_l + 1} \rfloor \tag{NC2}$$

And the sufficient condition is that

$$d_i \leq \lfloor \frac{n_1 d_1}{d_1 + 1} \rfloor + \lfloor \frac{n_2 d_2}{d_2 + 1} \rfloor + \cdots + \lfloor \frac{n_i d_i}{d_i + 1} \rfloor, \ \forall i \in [1, l] \tag{SC}$$

We note that these conditions and the realizing graphs, when any exists, can be all computed in polynomial time. [7]

For the special case of l bounded by 3, it holds that the profile σ is MinNDeg realizable if and only if along with NC1 and NC2, the next condition is also satisfied.

$$d_2 \leq \lfloor \frac{n_1 d_1}{d_1 + 1} \rfloor + \lfloor \frac{n_2 d_2}{d_2 + 1} \rfloor, \ \textit{or} \ d_3 + 1 \leq n_1 + n_2 + n_3 - (1 + \lceil \frac{d_2 - n_2}{d_1} \rceil) \tag{NC3}$$

Acyclic Realization : When the required graph is acyclic, then for any sequence σ holds that

$$\phi(\sigma) = d_l^2 + 1 + \sum_{i=1}^{l}(n_i - 1)(d_i - 1)^2 + \sum_{i=1}^{l-1} d_i(d_i - 1)$$

Also, a sequence σ is MinNDeg realizable by a tree if and only if both of the following necessary conditions are true.

$$d_1 = 1 \textit{ and } \phi(\sigma) \leq n(\sigma) \tag{NC-Tree}$$

We recall here that $n(\sigma) = \sum_{i=1}^{l} n_i$. Observe that when we have the profile (1^n), condition NC-Tree amounts to claiming that (1^n) is realizable for every $n \geq 2$. Indeed, the star graph provides such a realization.

Now, if we want $\phi(\sigma)$ to be independent of d_1 and n_1 when $l > 1$, but we know that n_1 appears to $n(\sigma)$, then the condition NC-Tree can be rewritten as $\phi(\sigma) - \sum_{i=2}^{l} n_i \leq n_1$. So, the left side is absolutely independent of d_1 and n_1. Therefore, for any sub-profile, say $\sigma' = (d_l^{n_l}, \ldots, d_2^{n_2})$ of σ can be realized if it is expanded into a full profile $\sigma = (d_l^{n_l}, \ldots, d_1^{n_1})$ for which n_1 is large enough. So, in a sense, these n_1 vertices are leaves or neighbors of leaves, and control the realizability of the profile.[7]

5.1 Realizations on acyclic graphs

In this section we give a full characterisation for a profile to be realizable by acyclic graphs.

5.1.1 The main algorithm

Proposition 5.2. *[7] Any sequence $\sigma = (d_l^{n_l}, \ldots, d_1^{n_1})$ satisfying $d_1 = 1$ and $\phi(\sigma) \leq n(\sigma)$ is* MinNDeg *realizable over trees.*

Proof. Let H be a star with a root r and d_l leaves. Denote the first set of $d_l + 1$ vertices as X_0. Holds that $|X_0 \setminus \{r\}| = d_l > l - 1$, since $d_1 = 1$. Now, we separate $X_0 \setminus \{r\}$ into 2 different sets, let Z_1 and Z_2 of size $l - 1$ and $d_l - l + 1$ respectively. We label the $(i-1)^{th}$ vertex in Z_1 as $v_{i,1}, \forall i \in [2, l]$. Note that $|Z_2| \geq 1$.

The algorithm below builds H in l rounds, $i = l, \ldots, 1$. (See Algorithm 4 for pseudocode).

Corollary 5.1. *Before the beginning of round i, we set the vertex $v_{i,1}$ to be a leaf node in the partially constructed tree H and its neighbor r has always degree at least $d_l \geq d_i$.*

Now, we are ready to give the description of rounds of the algorithm.

Description of round i $(i > 1)$: Take the leaf $v_{i,1} \in X_0$. Add $n_i - 1$ new vertices, say $v_{i,2}, \ldots, v_{i,n_i}$ and connect each $v_{i,j}$ to $v_{i,j-1}$, for $2 \leq j \leq n_i$. Let V_i denote the set $\{v_{i,1}, \ldots, v_{i,n_i}\}$. Note that V_i forms a simple path, as we can see in the Figure 5.1. Note also that by our corollary, the neighbor of $v_{i,1}$ (other than $v_{i,2}$ in H) had degree at least d_i. So, we take the following conditions.

1. All vertices in V_i have degree d_i.

2. All neighbors of vertices in V_i have degree at least d_i.

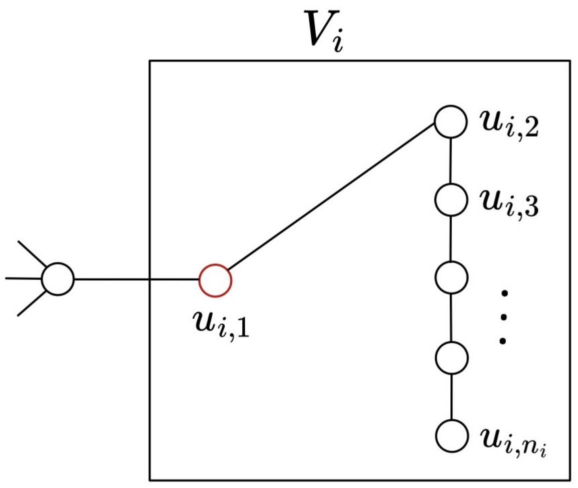


Figure 5.1: An example that realizes the description of round i.

In order to ensure condition 1, we move on as follows: (i) Since the vertices $v_{i,1}, \dots, v_{i,n_i-1}$ have degree 2 in the H_i, they are connected to $d_i - 2$ new vertices and (ii) the vertex v_{i,n_i} is connected to $d_i - 1$ new vertices. In the process, we add $n_i(d_i - 2) + 1$ new vertices, and we denote them as A_i.

In order to ensure condition 2, we connect each $a \in A_i$ to $d_i - 1$ new vertices. Let B_i be the set of these new vertices. Then, $|B_i| = |A_i| \cdot (d_i - 1)$.

We need to compute the size of $V_i \cup A_i \cup B_i$. So, we have that,

$$\begin{aligned}|V_i \cup A_i \cup B_i| &= n_i + (n_i(d_i - 2) + 1) + (n_i(d_i - 2) + 1)(d_i - 1)\\ &= n_i + d_i(n_i(d_i - 2) + 1)\\ &= n_i(d_i - 1)^2 + d_i\end{aligned}$$

Description of round 1 : Add a set Y_0 of $n(\sigma) - \phi(\sigma)$ new vertices to H. We can observe that $n(\sigma) - \phi(\sigma) \geq 0$ thanks to the assumption. Connect the root node r in H to each of the vertices in Y_0. We need to prove that holds $|V(H)| = n(\sigma)$.

$$\begin{aligned}|V(H)| &= |X_0| + \sum_{i=2}^{l}(|V_i \cup A_i \cup B_i| - 1) + |Y_0|\\ &= d_l + 1 + \sum_{i=2}^{l}[n_i(d_i - 1)^2 + d_i - 1] + n(\sigma) - \phi(\sigma)\\ &= d_l^2 + 1 + \sum_{i=2}^{l}(n_i - 1)(d_i - 1)^2 + \sum_{i=2}^{l-1} d_i(d_i - 1) + n(\sigma) - \phi(\sigma)\\ &= n(\sigma)\end{aligned}$$

Correctness analysis Let V_1 be the set $V(H) \setminus \cup_{i=2}^{l} V_i$. It is obvious that $|V_i| = n_i, \forall i \in [2, l]$. Also, since $|V(H)| = n(\sigma)$ it results that $|V_1| = n(\sigma) - \sum_{i=2}^{l} n_i = n_1$. So, we need to show that $\forall u \in V_i, \textsc{MinNDeg}(u) = d_i, \forall i \in [1, l]$. Observe that the degrees of vertices in $V_i \cup A_i$ don't change after round i, so conditions 1 and 2 we saw before, continue to apply for every $V_i, \forall i \in [2, l]$. This ensures us that for any $u \in V_i, \textsc{MinNDeg}(u) = d_i, \forall i \in [2, l]$. We need now analyse the set V_1. More specifically, we have:

$$V_1 = (X_0 \setminus \cup_{i=2}^{l}\{v_{i,1}\}) \cup Y_0 \cup (\cup_{i=2}^{l}(A_i \cup B_i)) = \{r\} \cup Z_2 \cup Y_0 \cup (\textstyle\sum_{i=2}^{l}(A_i \cup B_i))$$

For $2 \leq i \leq l$ the set B_i contains only leaves and each node in A_i must have a neighbor in B_i. So, the vertices in $\cup_{i=2}^{l}(A_i \cup B_i)$ have $\textsc{MinNDeg} = 1$.

Also, we need to check the vertices of $\{r\} \cup Z_2 \cup Y_0$, of which the vertices in $Z_2 \cup Y_0$ have already degree 1. However, r is adjacent to vertices with degree 1, in Z_2, so MinNDeg$(r) = 1$.

□

Algorithm 4: Computing a tree MinNDeg realization for a profile σ.

```
   input : A sequence $\sigma = (d_l^{n_l}, \dots, d_1^{n_1})$ satisfying $d_1 = 1$ and
           $n(\sigma) \geq \phi(\sigma)$.
1  Initialize $H$ to be a star with a root $r$ and $d_l$ leaves.
2  Label the $i^{th}$ leaf in $H$ as $v_{i,1}, \forall i \in [2, l]$.
3  for $i = l$ to 2 do
4      Add $n_i - 1$ new vertices to $H$, namely $v_{i,2}, \dots, v_{i,n_i}$.
5      Connect each $v_{i,j}$ to $v_{i,j-1}, \forall 2 \leq j \leq n_i$.
6      Add to $H$ a set $A_i$ of $n_i \cdot (d_i - 2) + 1$ new vertices.
7      Connect each $v_{i,j}, \forall 1 \leq j \leq n_i - 1$, to $d_i - 2$ isolated vertices in $A_i$.
8      Connect $v_{i,n_i}$ to $d_i - 1$ isolated vertices in $A_i$.
9      Add to $H$ a set $B_i$ of $|A_i| \cdot (d_i - 1)$ new vertices.
10     Connect each $a \in A_i$ to $d_i - 1$ isolated vertices in $B_i$.
11 end
12 Add $n(\sigma) - \phi(\sigma)$ new vertices to $H$ as children of the root $r$.
   output: $H$.
```

5.1.2 Tightness criterion

In this section, we want to show that the above construction is tight. For example, a sequence is MinNDeg realizable by trees if and only if it is realizable by the procedure we described in Proposition 5.2.

Proposition 5.3. *[7] For a sequence $\sigma = (d_l^{n_l}, \dots, d_1^{n_1})$ satisfying $d_1 = 1$ a necessary condition to be* MinNDeg *realizable over trees is $\phi(\sigma) \leq n(\sigma)$.*

Proof. Take any profile $\sigma = (d_l^{n_l}, \dots, d_1^{n_1})$, and let T be a MinNDeg tree realization of the profile on V. Let $r \in V(T)$ be a vertex that satisfies MinNDeg$(r) = d_l$. Let also node r be the root of the graph T. For $i = 1, \dots, l$, let $V_i = \{v \in V(T) \mid \text{MinNDeg}(v) = d_i\}$. We can see that for every $i < l$, there exists at least one edge, denoted $(y_i, x_i) \in E(T)$, where y_i is the parent of x_i, satisfying the next conditions. First of all, $x_i \in V_i$ and also none of the vertices in the tree path $r \to T\ y_i$ lie in V_i. These edges have a very important role in the tight bound on $\phi(\sigma)$. We denote,

$$A = \{x_i \mid \textsc{MinNDeg}(y_i) < d_i,\ for\ i < l\},$$
$$B = \{x_i \mid \textsc{MinNDeg}(y_i) > d_i,\ for\ i < l\}.$$

For every $w \in V(T)$, let $\mathcal{C}_w$ and $\mathcal{GC}_w$ denote the set of the children and grand-children of w in T, respectively. Moreover, $\mathcal{C}_A = \cup_{w \in A} \mathcal{C}_w$.

Now, we are able to define a function $\Gamma : V \to 2^V$ (see Figure 5.2 for an example).

$$\Gamma(w) = \begin{cases} \{r\} \cup \mathcal{C}_r \cup \mathcal{GC}_r, & w = r \\ \mathcal{C}_w \cup (\mathcal{GC}_w \setminus \mathcal{C}_A), & w \in A \\ \mathcal{GC}_w \setminus \mathcal{C}_A, & otherwise \end{cases}$$

Figure 5.3 realizes the sub-tree which is caused by $\{w\} \cup \mathcal{C}_w \cup \mathcal{GC}_w$, for any vertex w.

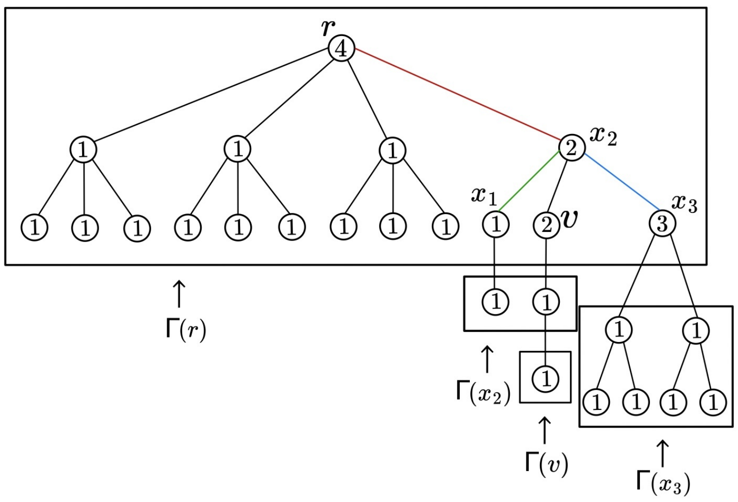


Figure 5.2: An example of a tree MinNDeg realization for the profile $\sigma = (4^1, 3^1, 2^2, 1^{22})$. The number in every node denotes its MinNDeg. Also, the edges (y_1, x_1), (y_2, x_2) and (y_3, x_3) are colored green, red and blue respectively. Moreover, $A = \{x_3\}$ and $B = \{x_1, x_2\}$.

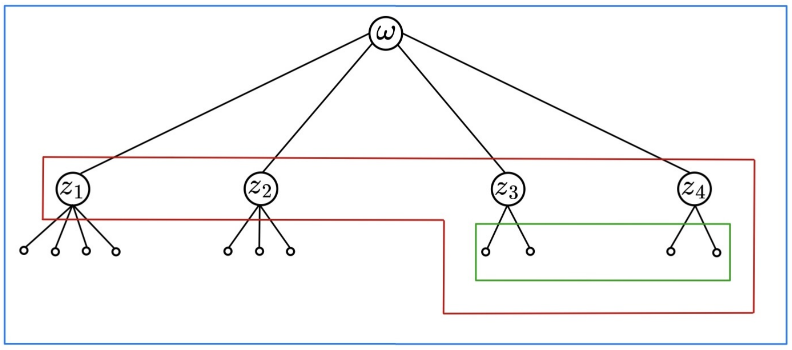


Figure 5.3: $\Gamma(w)$ is the set of vertices in the blue line if $w = r$, in the red one, if $w \in A$ and finally in the green line otherwise. Assuming that $z_1, z_2 \in A$.

Next, we will use 2 claims in order to prove the Proposition 5.3.

Claim 5.1. *[7] $\Gamma(w) \cap \Gamma(v) = \emptyset$ for any $v, w \in V(T)$ such that $v \neq w$.*

Proof. We first take the case $v = r$. If $w \notin \mathcal{C}_r$ or $w \notin A$, then the result holds. However, if $w \in \mathcal{C}_r$, then it is obvious that MinNDeg$(r) \geq$ MinNDeg(w), implying therefore that $w \notin A$.

Now, we assume that we have any 2 vertices, let $v \neq w \in V(T) \setminus \{r\}$. We consider to the contrary that $\Gamma(v) \cap \Gamma(w)$ contains a vertex, say z. Then z must be a child of 1 of the vertices v or w and the corresponding node must lie in A. We take the scenario that $z \in \mathcal{C}_w$ and $w \in A$. Since $z \in \mathcal{C}_w \subseteq \mathcal{C}_A$, then $z \notin \mathcal{GC}_v \setminus \mathcal{C}_A$. So z cannot be a child of v, implying that $z \notin \Gamma(v)$. So, finally we have that $\Gamma(v) \cap \Gamma(w)$ must be empty. □

Claim 5.2. *[7] For any $v \in V_i$, $1 \leq i \leq l$, it is true the following,*

$$|\Gamma(v)| \geq \begin{cases} d_i(d_i - 1), & \textit{if } v \in A \cup B \\ (d_i - 1)^2, & \textit{if } v \notin \{r\} \cup A \cup B \end{cases}$$

Proof. We take a vertex $v \in V_i$, for any $i \leq l$. We can see that each $u \in \mathcal{C}_v$ must have degree at least d_i and satisfy the condition $|\mathcal{C}_u| \geq d_i - 1$. Now, let z_0 be $v's$ parent and $z_1, \ldots, z_t$ be the vertices in $\mathcal{C}_v \cap A$. Since z_0 is a progenitor of $z_1, \ldots, z_t$, then by definition of $A \cup B$, the MinNDeg of all the vertices $z_0, z_1, \ldots, z_t$ must be distinct.

Without loss of generality, we can consider that $\text{MinNDeg}(z_t) > \cdots > \text{MinNDeg}(z_1)$. So, by definition of A, holds that $\text{MinNDeg}(z_1) > d_i$. We now define a new variable, let $\Delta = max_{j=0}^{t}\text{MinNDeg}(z_j)$. Therefore, we have $deg(v) \geq \Delta$. Also, since $\Delta \geq \text{MinNDeg}(z_t) > \cdots > \text{MinNDeg}(z_1) > d_i$, then we have that $\Delta \geq d_l + t$. Therefore,

$$|\mathcal{C}_v \setminus A| = deg(v) - t - 1 \geq \Delta - t - 1 \geq (d_i - 1) \tag{5.1}$$

We now are able to consider 3 different cases according to v, i.e. whether v lies in A or B or $V \setminus (\{r\} \cup A \cup B)$.

1. If $v \in A$: By Equation 5.1, we have $|\mathcal{GC}_v \setminus \mathcal{C}_A| \geq (d_i - 1)^2$ and $|\mathcal{C}_v| \geq d_i - 1$. Combined, we take that $|\Gamma(v)| = |\mathcal{GC}_v \setminus \mathcal{C}_A| + |\mathcal{C}_v| \geq d_i(d_i - 1)$.

2. If $v \in B$: Using the definition of B, we have that $\text{MinNDeg}(z_0) > d_i$. So, $\text{MinNDeg}(z_j) > d_i$, for any $j \in [0, t]$. Also, it holds that MinNDeg of $z_0, \ldots, z_t$ are distinct. Therefore, $\Delta \geq d_i + (t + 1)$ and $|\mathcal{C}_v \setminus A| = deg(v) - t \geq \Delta - t \geq d_i$. Implying that $|\Gamma(v)| = |\mathcal{GC}_v \setminus \mathcal{C}_A| \geq d_i(d_i - 1)$.

3. If $v \notin \{r\} \cup A \cup B$: By Equation 5.1, we have $|\mathcal{GC}_v \setminus \mathcal{C}_A| \geq (d_i - 1)^2$. This implies that $|\Gamma(v)| \geq (d_i - 1)^2$.

And now the claim follows. □

We note that $\Gamma(r)$ contains at least $d_l^2 + 1$ vertices, because the degrees of r and of its children are at least d_l. Now, we are able to prove the bound over $\phi(\sigma)$. For simplicity, we will denote the vertex r, as x_l.

$$\begin{aligned} n(\sigma) = |V(T)| \geq |\Gamma(r)| + \sum_{i=1}^{l} \sum_{v \in V_i \setminus \{x_i\}} |\Gamma(v)| + \sum_{i=1}^{l-1} |\Gamma(x_i)| \\ \geq d_l^2 + 1 + \sum_{i=1}^{l} (n_i - 1)(d_i - 1)^2 + \sum_{i=1}^{l-1} d_i(d_i - 1) = \phi(\sigma) \end{aligned}$$

The proof is complete now. □

Corollary 5.2. *For a sequence $\sigma = (d_l^{n_l}, \ldots, d_1^{n_1})$ satisfying $d_1 = 1$ is* MinNDeg *realizable over forests if holds $\phi(\sigma) \leq n(\sigma)$.*

Proof. If we have a sequence which is MinNDeg realizable as a forest, then we can split it into 2 sub-sequences $\sigma_1, \ldots, \sigma_k$ corresponding to each of its connected components. By Proposition 5.3, we have that $n(\sigma_i) \geq \phi(\sigma_i), \forall i \in [1, k]$. So, $n(\sigma) = \sum_{i=1}^{k} n(\sigma_i) \geq \sum_{i=1}^{k} \phi(\sigma_i) \geq \phi(\sigma)$. It is easy to verify that the last inequality follows from the definition of ϕ. □

It is true that a tree always contains vertices of degree 1. So,by Corollary 5.2 and Proposition 5.2, and from this fact, results the following theorem.

Theorem 5.1. *[7] A sequence $\sigma = (d_l^{n_l}, \ldots, d_1^{n_1})$ is* MinNDeg *realizable over acyclic graphs if and only if hold the following conditions.*

1. *$d_1 = 1$ and*
2. *$\phi(\sigma) \leq n(\sigma)$.*

5.2 Realizations in General graphs

In this section we will see some new definitions, so we need to define them first, for our construction. Therefore, let $G = (V, E)$ be any graph. For some vertex $v \in V$, we define $leader(v)$ to be a vertex in $N[v]$ of minimum degree and if there is more than one choice we pick the leader arbitrarily. Note, these arbitrarily chosen leaders need not be consistent across neighbors. For example, it is possible that 2 vertices, let w and u, are the leaders of each other. In other words, $leader(u) \in arg\ min\{deg(v) | v \in N[u]\}$.

Next, let $\sigma = (d_l^{n_l}, \ldots, d_1^{n_1})$ be the minimum degree sequence of G. So, let V_i denote the set of those vertices in G whose minimum degree in the closed neighborhood is d_i, and then we have that $|V_i| = n_i$. Moreover, let L_i be the set of those vertices in G who are leaders of at least one vertex in V_i, equivalently, $L_i = \{leader(v) | v \in V_i\}$ and denote by $L = \cup_{i=1}^{l} L_i$ the set of all the leaders in the graph. We can see that the sets $V_1, \ldots, V_l$ form a partition of the vertex set of G.

Now, we have to define the follower. So, a vertex v in G is said to be a follower, if $leader(v) \neq v$. Let, $F_i = \{v \in V_i | v \neq leader(v)\}$ be the set of all the followers in V_i. Additionally, we need to define $R = V \setminus L$. This is the set of all the non leaders. Finally, we define $F = \cup_{i=1}^{l} F_i$ to be the set of all the followers.

in Figure 5.4 we give an example for the definitions we talked about.

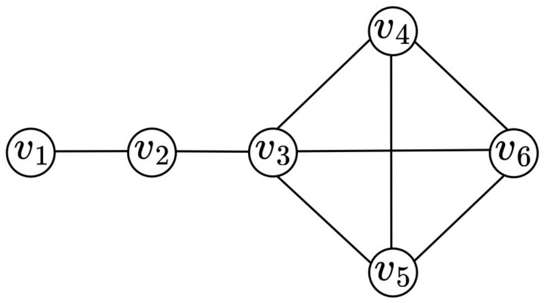


Figure 5.4: A MinNDeg realizable graph for the sequence $\sigma = (3^3, 2^1, 1^2)$.

In Figure 5.4 holds that MinNDeg(v_1) = MinNDeg(v_2) = $deg(v_1) = 1$, MinNDeg(v_3) = $deg(v_2) = 2$ and MinNDeg$(v_i) = 3$, $\forall i \in \{4, 5, 6\}$. It is obvious that $leader(v_2) = v_1$ and $leader(v_3) = v_2$. So, v_2 is a leader and a follower as well.

Our goal here is that there exist realizable sequences σ for which any graph G realizing σ and any leader function over G, the sets L and F have non empty intersection. For instance, assume the sequence in Figure 5.4, $\sigma = (3^3, 2^1, 1^2)$. It is easy to verify that σ has only 1 realizing graph and in it, the leader and follower sets are non disjoint.[7]

So, we can sort the sequences that accept disjoint leader and follower sets as follows.

Definition 5.1. *A sequence $\sigma = (d_l^{n_l}, \ldots, d_1^{n_1})$ is said to admit a Disjoint $Leader-Follower$ (called* DLF*)* MinNDeg *realization if there exists a graph G realizing σ and a leader function under which the sets L and F are mutually disjoint, i.e., $L \cap F = \emptyset$.*

Theorem 5.2. *[7] For every $\sigma = (d_l^{n_l}, \ldots, d_1^{n_1})$ that is* MinNDeg *realizable* *by a graph, let G, must be met the following necessary conditions.*

$$d_i \le (\sum_{j=1}^{i} n_j) - 1, \forall i \in [1,l] \tag{NC1}$$

$$d_l \le \sum_{j=1}^{l} \lfloor \frac{n_j d_j}{d_j + 1} \rfloor \tag{NC2}$$

Moreover, for every leader function defined over G and $i < l$, if $L_i \cap V_i \neq \emptyset$, then

$$d_i \le \sum_{j=1}^{i} \lfloor \frac{n_j d_j}{d_j + 1} \rfloor .$$

Proof. Firstly, we give a lower bound on the size of the leader set L_i. Note that for every $i \in [1,l]$, holds $|L_i| \ge \lceil \frac{n_i}{d_i + 1} \rceil$. Next, we take a vertex $a \in L_i$. Since $|N[a]| = d_i + 1$, then the vertex a can be considered as leader for at most $d_i + 1$ vertices. This can verify that $|L_i| \ge \frac{n_i}{d_i+1}$. So, since $|L_i|$ is integer, the claim arises.

Proof of $(NC1)$. Let w be any vertex in G such that $deg(w) = d_i$. Then w and all the neighbors of w must be contained in $\cup_{j=1}^{i} V_j$, and so we take the following.

$$d_i + 1 = |N[w]| \le | \cup_{j=1}^{i} V_j| = \sum_{j=1}^{i} n_j$$

And this proves the condition (NC1).

Proof of $(NC2)$. We consider the case where w is a vertex in G such that $\textsc{MinNDeg}(w) = d_l$. Therefore, $N[w]$ can not contain vertices of degree less than d_l and follows that $N[w] \cap L_i = \emptyset$, $\forall i < l$. So, $|N[w]| \le n - \sum_{i=1}^{l-1} |L_i|$. Moreover, the degree of the vertex w must be at least d_l. Finally, we take the following.

$$d_l + 1 \le |N[w]| \le n - \sum_{i=1}^{l-1} |L_i| = n_l + \sum_{i=1}^{l-1} (n_i - |L_i|) \le n_l + \sum_{i=1}^{l} \lfloor \frac{n_i d_i}{d_i + 1} \rfloor$$

If we consider the case $n_l \le d_l$, then we have $n_l - 1 = \lfloor \frac{n_l d_l}{d_l+1} \rfloor$ and therefore $d_l \le \sum_{i=1}^{l} \lfloor \frac{n_i d_i}{d_i+1} \rfloor$. Now if $n_l \ge d_l + 1$ then $\frac{n_l d_l}{d_l+1} \ge d_l$ and follows that $\lfloor \frac{n_l d_l}{d_l+1} \rfloor \ge d_l$, because d_l is integer.

Proof of the last claim. We consider a vertex lying in $L_i \cap V_i$, say w, and we have that $\textsc{MinNDeg}(w) = deg(w) = d_i$. Note that for any $j < i$, the vertices in the set L_j have degree strictly less than d_i. We know that $N[w]$ can not contain vertices of degree less than d_i, so for every $j < i$, $N[w] \cap L_j = \emptyset$. Moreover, vertices in $V_{i+1} \cup \cdots \cup V_l$ can not be adjacent to any vertex in $\{w\} \cup (\cup_{j=1}^{i-1} L_j)$, so we have that $N[w]$ and $\cup_{j=1}^{i-1} L_j$ are both contained in $\cup_{j=1}^{i} V_j$. Finally, we take the following.

$$d_i+1 = |N[w]| \leq |\cup_{j=1}^{i} V_j| - |\cup_{j=1}^{i-1} L_i| = n_i + \textstyle\sum_{j=1}^{i-1}(n_i - |L_j|) \leq n_i + \sum_{j=1}^{i-1} \lfloor \frac{n_j d_j}{d_j+1} \rfloor$$

If $n_i < d_i$, then holds $n_i - 1 = n_i - \lceil \dfrac{n_i}{d_i+1} \rceil = \lfloor \dfrac{n_i d_i}{d_i+1} \rfloor$ and therefore $d_i \leq \sum_{j=1}^{i} \lfloor \dfrac{n_j d_j}{d_j+1} \rfloor$.

Now, if we consider the case $n_i \geq d_i + 1 \iff n_i \cdot d_i \geq d_i \cdot (d_i+1) \iff \dfrac{n_i \cdot d_i}{d_i+1} \geq d_i$. Since d_i is integral results $\lfloor \dfrac{n_i d_i}{d_i+1} \rfloor \geq d_i$.

□

Theorem 5.3. *[7] Every sequence $\sigma = (d_l^{n_l}, \ldots, d_1^{n_1})$ satisfying the sufficient condition,*

$$d_i \leq \sum_{j=1}^{i} \lfloor \frac{n_j d_j}{d_j+1} \rfloor, \ \forall i \in [1, l] \qquad \text{(SC)}$$

is MinNDeg *realizable. Also, we can compute a realizing graph, let G, and a leader function defined over G that satisfies the condition $L \cap F = \emptyset$.*

Proof. First of all, we will consider the most simple case, the realizability of uniform sequences and after that, we will see the case of general sequences.

Uniform Sequences. Firstly, we assume the sequence $\sigma = (d^n)$. If $n \geq d+1$, we can have a realization for the sequence σ. We consider 2 integers, let $q \geq 1$ and $r \in [0, d]$, satisfying $n = q \cdot (d+1) - r$. Next take a set, say A of q vertices, namely a_i, where $i \in [1, q]$ and another set, say B of d_q vertices, namely b_{ij}, where $i \in [1, q]$ and $j \in [1, d]$. Connect each vertex a_i to $b_{i1}, ..., b_{id}$. Therefore, the vertices in A will have degree d and the vertices in B will have in their neighborhood a vertex of degree exactly d.

Now, if $r > 0$, then we have to merge b_{1j} with b_{2j}, $\forall j \in [1, r]$, thus decreasing r vertices in B. Recall that b_{1j} and b_{2j} exists since $r > 0$ only when $q \geq 2$. So, $|A| + |B| = n$ and every vertex in A still has degree d. Thus we have the following.

$$|A| = \frac{n+r}{d+1} = \lceil \frac{n}{d+1} \rceil \; and \; |B| = n - |A| = \lfloor \frac{nd}{d+1} \rfloor \geq d$$

The last step is to add edges between each pair of vertices in B in order to make it a clique of size at least d. This entail that the vertices in set B will have degree at least d. It is easy to verify that MinNDeg(v), $\forall v \in A \cup B$ is d. In our constructed graph the set A is the leader set, and the set B is the follower set.

In the rest of proof, we will denote as Graph(n, d, A, B) a function that returns the edges of the graph as constructed above whenever provided with four parameters n, d, A, B satisfying $n \geq d + 1$, $|A| = \lceil \frac{n}{d+1} \rceil$ and $|B| = \lfloor \frac{nd}{d+1} \rfloor$.

General Sequences. We take the case $\sigma = (d_l^{n_l}, \ldots, d_1^{n_1})$. First of all, we consider that G is an empty graph. Our algorithm uses l rounds. (Refer to Algorithm 5.) In every round, we add to G a set V_i of n_i new vertices and partition V_i into 2 sets, let L_i with size $\lceil \frac{n_i}{d_i+1} \rceil$ and R_i with size $\lfloor \frac{n_i d_i}{d_i+1} \rfloor$. If $n_i > d_i + 1$, then we can solve this round by adding to G all the edges returned by Graph(n_i, d_i, L_i, R_i). Note that if $n_i \leq d_i + 1$, then the set L_i will contain only one vertex, let a_i. In this case, we need to add edges between a_i and all the vertices in R_i. Moreover, we add edges between a_i and any arbitrarily chosen $d_i + 1 - n_i$ vertices in $\cup_{j<i} R_j$. This can be possible since holds the following.

$$d_i + 1 - n_i = d_i - \lfloor \frac{n_i d_i}{d_i + 1} \rfloor \leq \sum_{j=1}^{i-1} \lfloor \frac{n_j d_j}{d_j + 1} \rfloor = \sum_{j=1}^{i-1} |R_j|$$

After the l rounds are completed, we finally add edges between each pair of vertices in $R = \cup_{i=1}^{l} R_i$, in order to make it a clique.

Now, we are able to take bounds on the degree of vertices in L_i and R_i.

1. Every vertex in L_i has degree d_i. Note that we add edges to vertices in L_i only in the i^{th} iteration of the for loop. If $n_i > d_i + 1$, then the degree of each vertex in L_i is d_i. Now, if $|L_i| = 1$ or $n_i \leq d_i + 1$, then

$|R_i| = n_i - |L_i| = n_i - 1$ and therefore the degree of the vertex $a_i \in L_i$ is $(n_i - 1) + (d_i + 1 - n_i) = d_i$.

2. Every vertex in R_i has degree at least d_l. For every $i \in [1, l]$, if holds that $n_i > d_i + 1$, then $|R_i| = \lceil \frac{n_i d_i}{d_i+1} \rceil$, and even in the case $n_i \leq d_i + 1$, we take $|R_i| = n_i - |L_i| = n_i - \lceil \frac{n_i}{d_i+1} \rceil = \lceil \frac{n_i d_i}{d_i+1} \rceil$. So, we have that $|R| = \sum_{i=1}^{l} |R_i| = \sum_{i=1}^{l} \lceil \frac{n_i d_i}{d_i+1} \rceil$ which is bounded below by d_i. It is true that every vertex in R is adjacent to at least 1 vertex in $\cup_i L_i$ and since $|R| \geq d_l$, then the degree of vertices in R is at least d_l.

Next, we show that for any vertex $v \in V_i$, MinNDeg$(v) = d_i$, where $i \in [1, l]$. We will consider 2 cases.

- If $v \in L_i$, then MinNDeg$(v) = d_i$, because every vertex in L_i has degree d_i and is adjacent to only vertices in R which have degree at least $d_l \geq d_i$.
- If $v \in R_i$, then MinNDeg$(v) = d_i$, also because every vertex in R_i is adjacent to at least 1 vertex in L_i and $N[v]$ is contained in the set $R \cup (\cup_{j \geq i} L_j)$ and its vertices have degree at least d_i.

The leader function over V constructs as follows. For every vertex $v \in \cup_{i=1}^{l} L_i$ we set $leader(v) = v$ and for every $v \in R_i$ we set $leader(v)$ to be any neighbor of v in L_i. It is true that for every vertex in $L = \cup_{i=1}^{l} L_i = \{leader(v) \mid v \in V\}$ is a leader of itself. So, the sets L and R which are leader and follower respectively, must be disjoint.

□

The following theorem is a consequence of the above results.

Theorem 5.4. *[7] The sequence $\sigma = (d_2^{n_2}, d_1^{n_1})$ is* MinNDeg *realizable if and only if $d_1 \leq \lfloor \frac{n_1 d_1}{d_1+1} \rfloor$ and $d_2 \leq \lfloor \frac{n_1 d_1}{d_1+1} \rfloor + \lfloor \frac{n_2 d_2}{d_2+1} \rfloor$.*

Proof. We first assume that σ is realizable. Then Theorem 5.2 implies the following.

1. $n_1 \geq d_1 + 1$ which implies $d_1 \leq \lfloor \frac{n_1 d_1}{d_1+1} \rfloor$.
2. $d_l = d_2 \leq \lfloor \frac{n_1 d_1}{d_1+1} \rfloor + \lfloor \frac{n_2 d_2}{d_2+1} \rfloor$.

Finally, the converse follows from Theorem 5.3. □

Algorithm 5: Computing a MinNDeg realization for a given special σ.

```
input : A sequence $\sigma = (d_l^{n_l}, \ldots, d_1^{n_1})$ satisfying $d_i \le \sum_{j=1}^{i} \lfloor \frac{n_j d_j}{d_j+1} \rfloor$ for
        every $1 \le i \le l$.
1  Initialize $G$ to be an empty graph.
2  for $i = 1$ to $l$ do
3      Add to $G$ a set $V_i$ of $n_i$ new vertices.
4      Partition $V_i$ in 2 sets $L_i$, $R_i$ such that $|L_i| = \lceil \frac{n_i}{d_i+1} \rceil$ and
        $|R_i| = \lfloor \frac{n_i d_i}{d_i+1} \rfloor$.
5      if ($n_i > d_i + 1$ or $|L_i| > 1$) then
6          Add to $G$ all the edges returned by GRAPH$(n_i, d_i, L_i, R_i)$.
7      end
8      if ($|L_i| = 1$) then
9          Let $a_i$ be the only vertex in $L_i$.
10         Connect $a_i$ to all vertices in $R_i$ and any arbitrary $d_i + 1 - n_i$
             vertices in $\cup_{j<i} R_j$.
11     end
12 end
13 Add edges between each pair of vertices in $R = \cup_{i=1}^{l} R_i$ to make it a
     clique.
output: $G$.
```

5.2.1 Sequences admitting disjoint Leader-Follower sets

Here, we will see a theorem that gives results on sequences admitting disjoint Leader-Follower sets.

Theorem 5.5. *[7] A sequence $\sigma = (d_l^{n_l}, \dots, d_1^{n_1})$ is* MinNDeg *realizable by a graph G having disjoint leader set, say L, and follower set, say F, with respect to some leader function, if and only if, for every $i \in [1, l]$, $d_i \leq \sum_{j=1}^{i} \lfloor \frac{n_j d_j}{d_j+1} \rfloor$.*

Proof. First of all, we consider that there exists a leader function over G for which $L \cap F = \emptyset$. Then, for every $i \in [1, l]$, $L_i \subseteq V_i$. This is true because if for some i, there exists $w \in L_i \setminus V_i$, then $deg(w) = d_i \neq$ MinNDeg(d_i), which implies that w is a leader and follower too. By Theorem 5.2 and by the fact that $L_i \subseteq V_i$, then $d_i \leq \sum_{j=1}^{i} \lfloor \frac{n_j d_j}{d_j+1} \rfloor$, $\forall i \in [1, l]$. By Theorem 5.3 results the converse claim. □

5.2.2 MinNDeg realization of tri-sequences in general graphs

In this section, we will take the case when a sequence has only three distinct degrees. In the following theorem is given a complete characterization of sequence $\sigma = (d_3^{n_3}, d_2^{n_2}, d_1^{n_1})$.

We will omit the proof of the Theorem 5.6, but there is on [7].

Theorem 5.6. *[7] The necessary and sufficient conditions for a sequence $\sigma = (d_l^{n_l}, \dots, d_1^{n_1})$, when $l = 3$, to be* MinNDeg *realizable, is the following.*

1. $d_1 + 1 \leq n_1$
2. $d_2 + 1 \leq n_1 + n_2$
3. $d_3 \leq \lfloor \frac{n_1 d_1}{d_1+1} \rfloor + \lfloor \frac{n_2 d_2}{d_2+1} \rfloor + \lfloor \frac{n_3 d_3}{d_3+1} \rfloor$
4. *either* $d_2 \leq \lfloor \frac{n_1 d_1}{d_1+1} \rfloor + \lfloor \frac{n_2 d_2}{d_2+1} \rfloor$ *or* $d_3 + 1 \leq n_1 + n_2 + n_3 - (1 + \lceil \frac{d_2 - n_2}{d_1} \rceil)$

CHAPTER 6

Conclusions

6.1 Tables of Computational Complexity

In this section we summarize the results of computational complexity, for the different problems and the different classes of graphs, many of which we mentioned in the previous chapters. The results are taken from the literature and are reported exactly in the tables.

More specifically we give the results in separate tables based on the type of graph, i.e. whether it is weighted or unweighted.

Table 6.1: Results for realization with unweighted graphs.

Graph family	Range-DR	Set-DR
General	2-Range-DR is NP-hard[11]	2-Set-DR is NP-hard[11]
	1-Range-DR is polynomial[24]	1-Set-DR is polynomial[24]
Tree	3-Range-DR is NP-hard[12]	2-Set-DR is NP-hard[11]
	1-Range-DR is polynomial[24]	1-Set-DR is polynomial[24]
Star	Range-DR is polynomial[11]	Set-DR is polynomial[11]
Path	2-Range-DR is polynomial[11]	2-Set-DR is polynomial[11]
	Range-DR is NP-hard[11]	5-Set-DR is NP-hard[11]
Cycle	2-Range-DR is polynomial[11]	2-Set-DR is polynomial[11]
	Range-DR is NP-hard[11]	5-Set-DR is NP-hard[11]

Table 6.2: Results for realization with weighted graphs.

Graph family	Range-DR	Set-DR
General	Range-DR is polynomial[29]	Open problem
Tree	3-Range-DR is NP-hard[12] 1-Range-DR is polynomial[6]	2-Set-DR is NP-hard[11] 1-Set-DR polynomial[6]
Star	Range-DR is polynomial[11]	2-Set-DR is polynomial[11] 6-Set-DR is NP-hard[11]
Path	2-Range-DR is polynomial[11] Range-DR is NP-hard[11]	2-Set-DR is polynomial[11] 5-Set-DR is NP-hard[11]
Cycle	2-Range-DR is polynomial[11] Range-DR is NP-hard[11]	2-Set-DR is polynomial[11] 5-Set-DR is NP-hard[11]

We also give below the complete characterization for the MaxNDeg realization problem for both cases the closed and the open neighborhood.

Table 6.3: MaxNDeg realization.

Graph family	Characterization
General	$d_l \leq n_l - 1,\ d_1 \geq 2$ or n_1 is even [10]
Connected	$d_l \leq n_l - 1,\ d_1 \geq 2$ or $\sigma = (1^2)$ [10]

It is interesting to contrast the behavior of the MinNDeg and MaxNDeg profiles. For general graphs, MinNDeg appears to be more difficult, since it is non monotone when edges are added or deleted, while the MaxNDeg profile is monotone. For trees, on the other hand, the realizability of the MinNDeg profile depends only on the leaves and their parents, which simplifies the analysis, but no a similar simplifying property was found for the MaxNDeg profile.

At this point, we present the table for MinNDeg realization problem.

Table 6.4: MaxNDeg$^-$ realization.

Graph family	Characterization
General	σ can be split into two profiles σ_1 and σ_2 such that (i) σ_1 has a connected MaxNDeg$^-$ realization, and (ii) $\sigma_2 = (1^{2a})$ or $\sigma_2 = (d^d, 1^{2a+1})$, for integers $d \geq 2,\ a \geq 0$ [10]
Connected	$d_l \leq min\{n_l, n-1\},\ d_1 \geq 2$ or $\sigma = (d^d, 1^1)$ or $\sigma = (1^2)$ $\sigma \neq (d_l^{d_l+1}, 2^1)$ [10]

Table 6.5: MinNDeg realization.

Graph family	Characterization
General	Open problem
Acyclic	A sequence σ can be MinNDeg realizable over trees if satisfies (i) $d_1 = 1$, and (ii) $\phi(\sigma) \leq n(\sigma)$ [7]

6.2 Open Problems

In the field of mathematics, theoretical computing is clearly a very important field and there are many applications even in our everyday life. Therefore, it stands to reason that there are many open problems that would be interesting to study.

In chapter 3, we introduce and study the Set Distance Realization problem, which is an extension of the Range Distance Realization problem. We study the computational complexity of k-Set-DR and k-Range-DR, as function of k, in various graph families.

It is clear that there are still many open questions, including the following.

- Range-DR in weighted general graphs can be solved in polynomial time, but the status of Set-DR is currently unclear.
- 3-Range-DR and 2-Set-DR are NP-hard in trees, but the status of the 2-Range-DR problem remains unsettled.
- For stars, the hardness of the k-Set-DR problem is unsettled for $k =$

$3, 4, 5$.

- For paths and cycles the k-Set-DR problem is unsettled for $k = 3, 4$.
- The status of Range-DR for paths and cycles is an open problem.

In chapters 4 and 5, we study the MaxNDeg and MinNDeg realization problems, which have similar logic but certainly MinNDeg seems to be more difficult in the case of general graphs. So the next question arises, which is also an open problem.

Open Question. Does there exist a closed form characterization for realizing MinNDeg profiles for general graphs?

Bibliography

[1] Adams, P., and Nikolayevsky, Y. Planar bipartite biregular degree sequences. *Discrete Mathematics 342*, 2 (2019), 433–440.

[2] Aigner, M., and Triesch, E. Realizability and uniqueness in graphs. *Discrete Mathematics 136*, 1-3 (1994), 3–20.

[3] Althöfer, I. On optimal realizations of finite metric spaces by graphs. *Discrete & computational geometry 3*, 2 (1988), 103–122.

[4] Arikati, S. R., and Maheshwari, A. Realizing degree sequences in parallel. *SIAM Journal on Discrete Mathematics 9*, 2 (1996), 317–338.

[5] Augustine, J., Choudhary, K., Cohen, A., Peleg, D., Sivasubramaniam, S., and Sourav, S. Distributed graph realizations. *IEEE transactions on parallel and distributed systems 33*, 6 (2021), 1321–1337.

[6] Baldisserri, A., et al. Buneman's theorem for trees with exactly n vertices. *arXiv preprint arXiv:1407.0048* (2014), 1.

[7] Bar-Noy, A., Choudhary, K., Cohen, A., Peleg, D., and Rawitz, D. Minimum neighboring degree realization in graphs and trees. In *28th Annual European Symposium on Algorithms (ESA 2020)* (2020), Schloss-Dagstuhl-Leibniz Zentrum für Informatik.

[8] Bar-Noy, A., Choudhary, K., Peleg, D., and Rawitz, D. Realizability of graph specifications: Characterizations and algorithms. In *International Colloquium on Structural Information and Communication Complexity* (2018), Springer, pp. 3–13.

[9] Bar-Noy, A., Choudhary, K., Peleg, D., and Rawitz, D. Graph profile realizations and applications to social networks. In *WALCOM: Algorithms and Computation: 13th International Conference, WALCOM 2019, Guwahati, India, February 27–March 2, 2019, Proceedings 13* (2019), Springer, pp. 3–14.

[10] Bar-Noy, A., Choudhary, K., Peleg, D., and Rawitz, D. Graph realizations: maximum degree in vertex neighborhoods. *Discrete Mathematics 346*, 9 (2023), 113483.

[11] Bar-Noy, A., Peleg, D., Perry, M., and Rawitz, D. Graph realization of distance sets. In *47th International Symposium on Mathematical Foundations of Computer Science (MFCS 2022)* (2022), Schloss-Dagstuhl-Leibniz Zentrum für Informatik.

[12] Bar-Noy, A., Peleg, D., Perry, M., and Rawitz, D. Composed degree-distance realizations of graphs. *Algorithmica 85*, 3 (2023), 665–687.

[13] Barrus, M. D., and Donovan, E. A. Neighborhood degree lists of graphs. *Discrete Mathematics 341*, 1 (2018), 175–183.

[14] Burstein, D., and Rubin, J. Sufficient conditions for graphicality of bidegree sequences. *SIAM Journal on Discrete Mathematics 31*, 1 (2017), 50–62.

[15] Chernyak, A. A., Chernyak, Z. A., and Tyshkevich, R. I. On forcibly hereditary p-graphical sequences. *Discrete mathematics 64*, 2-3 (1987), 111–128.

[16] Chung, F., Garrett, M., Graham, R., and Shallcross, D. Distance realization problems with applications to internet tomography. *Journal of Computer and System Sciences 63*, 3 (2001), 432–448.

[17] Culberson, J. C., and Rudnicki, P. A fast algorithm for constructing trees from distance matrices. *Information Processing Letters 30*, 4 (1989), 215–220.

[18] Dress, A. W. Trees, tight extensions of metric spaces, and the cohomological dimension of certain groups: a note on combinatorial properties of metric spaces. *Advances in Mathematics 53*, 3 (1984), 321–402.

[19] Erdös, P., and Gallai, T. Graphs with prescribed degrees of vertices (in hungarian). mat. lapok.

[20] Even, S., Itai, A., and Shamir, A. On the complexity of time table and multi-commodity flow problems. In *16th annual symposium on foundations of computer science (sfcs 1975)* (1975), IEEE, pp. 184–193.

[21] Feder, T., Meyerson, A., Motwani, R., O'Callaghan, L., and Panigrahy, R. Representing graph metrics with fewest edges. In *Annual Symposium on Theoretical Aspects of Computer Science* (2003), Springer, pp. 355–366.

[22] Gupta, G., Joshi, P., and Tripathi, A. Graphic sequences of trees and a problem of frobenius. *Czechoslovak Mathematical Journal 57* (2007), 49–52.

[23] Hakimi, S. L. On realizability of a set of integers as degrees of the vertices of a linear graph. i. *Journal of the Society for Industrial and Applied Mathematics 10*, 3 (1962), 496–506.

[24] Hakimi, S. L., and Yau, S. S. Distance matrix of a graph and its realizability. *Quarterly of applied mathematics 22*, 4 (1965), 305–317.

[25] Hammer, P. L., and Simeone, B. The splittance of a graph. *Combinatorica 1* (1981), 275–284.

[26] Havel, V. A remark on the existence of finite graphs. *Casopis Pest. Mat. 80* (1955), 477–480.

[27] Kim, H., Toroczkai, Z., Erdős, P. L., Miklós, I., and Székely, L. A. Degree-based graph construction. *Journal of Physics A: Mathematical and Theoretical 42*, 39 (2009), 392001.

[28] Mossel, E., and Ross, N. Shotgun assembly of labeled graphs. *IEEE Transactions on Network Science and Engineering 6*, 2 (2017), 145–157.

[29] Rubei, E. Weighted graphs with distances in given ranges. *Journal of Classification 33* (2016), 282–297.

[30] Schmeichel, E., and Hakimi, S. On planar graphical degree sequences. *SIAM Journal on Applied Mathematics 32*, 3 (1977), 598–609.

[31] Simoes-Pereira, J. An optimality criterion for graph embeddings of metrics. *SIAM journal on discrete mathematics 1*, 2 (1988), 223–229.

[32] Tyshkevich, R., Mel'nikov, O., and Kotov, V. Graphs and degree sequences: canonical decomposition. *Cybernetics 17*, 6 (1981), 722–727.

[33] VARONE, S. C. A constructive algorithm for realizing a distance matrix. *European journal of operational research 174*, 1 (2006), 102–111.

[34] ZVEROVICH, I. E., AND ZVEROVICH, V. E. Contributions to the theory of graphic sequences. *Discrete Mathematics 105*, 1-3 (1992), 293–303.