icteam
Institute of Information and Communication Technologies,
Electronics and Applied Mathematics

Topics in combinatorial matrix theory

Maguy Trefois

UCL
Université
catholique
de Louvain

Thesis submitted in partial fulfillment of the requirements for the degree of
Docteur en sciences

Dissertation committee:

Prof. Jean-Charles Delvenne (UCL, advisor)
Prof. Jean-Pierre Tignol (UCL, advisor)
Prof. François Glineur (UCL)
Prof. Paul Van Dooren (UCL)
Prof. Raphaël Vandebril (KULeuven, Belgium)
Prof. Philippe Lefèvre (UCL, chair)

February 2016

Acknowledgment

I still remember the day I arrived at the UCL. It was on 19 September 2005. At that time, I was already passionate about mathematics and especially about algebra, but I didn't know that this passion would keep me at the UCL for a wonderful 11-year journey through mathematics. The courses I followed during my studies confirmed indeed my passion. In particular, I have fond memories for the courses delivered by Jean-Pierre Tignol and for the master thesis we did together. Today, I would like to thank him for his teaching, which led me to the decision of starting a PhD.

In September 2010, I joined the department of applied mathematics as a new PhD student. There, I had the privilege of working with Jean-Charles, who was my mentor. Jean-Charles, you have remained over the years available and enthusiastic. I'm also very grateful to you for your constant support, our numerous meetings about science and for the multiple things you taught me to improve the quality of my research.

Jean-Charles, Jean-Pierre, François, Paul, Raphaël and Philippe, I really appreciated the time you have taken to read the draft of my thesis and to help me improve it. Your comments were very constructive and the present thesis would certainly not be the same without your positive remarks.

Paul, I would like also to warmly thank you for our collaboration and for sharing with me your high knowledge of matrices.

Of course, I do not forget my colleagues, from both the Euler and the Cyclotron buildings. You really make the department a pleasant place to work.

I extend special thanks to my officemates Emile, Nicolas and Florian. I had a really good time in the office with you.

To my friends, I would like to address a very big thank. Your support and the very good moments we spent together helped me overcome the hardships of research. There are so many of you who deserve special thanks that naming everyone would take many many pages. I'll just dare heartfelt thanks to Aurélie and to Stéphane, Thibault, Claude, Jean and Jofroi for our deep friendship that began a long long time ago.

A Jackie et Mimi, un tout grand merci pour les agréables moments passés à discuter ensemble.

Maman, papa, c'est aussi à vous que je dois aujourd'hui l'aboutissement de ce long parcours universitaire. Depuis toujours, vous m'avez soutenue dans l'accomplissement de mes projets et vous m'avez encouragée dans les moments

les plus difficiles. Sans vous, je ne serai sans doute pas arrivée là où je suis aujourd'hui. Merci.

A Magali, ma grande soeur. Jamais je n'oublierai tous les chouettes moments que nous avons passés ensemble: nos vacances, nos week-ends, nos soirées entre filles. Avec moi, tu as vécu tous les moments importants de la thèse. Merci d'avoir toujours été là.

Nonna e Nonnì, grazie per tutto quello che avete fatto. Non vi dimenticherò mai.

J'aimerais également remercier chaleureusement toutes les personnes qui me sont proches. Chacune à votre manière, vous m'avez apporté votre soutient.

Mes derniers remerciements, je les dédie à toi, Pierre-Alain. L'amour et le soutient que tu m'apportes chaque jour ne cessent de me rendre plus forte. A tes côtés, les petits soucis du quotidien s'effacent pour laisser place au bonheur. Partager ma vie avec toi est un plaisir qui, je l'espère, durera bien longtemps.

List of notation

Vectors and matrices

$\mathbb{R}$	Set of real numbers	
$\mathbb{R}^n$	Set of real column vectors of size n	
$\mathbb{R}^{m\times n}$	Set of real matrices with m rows and n columns	
I_n	Identity matrix of dimension n	
J_n	All ones matrix of size n	
$\mathbf{1}_n$	Column vector of all ones	
$D(p,n)$	De Bruijn matrix of order p and size n	87
$\mathrm{diag}(Q_1,...,Q_s)$	Block-diagonal matrix	96
a_{ij}	The (i,j)-entry of the matrix A (given by context)	
A^T	Transpose of the matrix A	
$\mathrm{rank}(A)$	Rank of the matrix A	
$\ker(A)$	Kernel of the matrix A	
$\dim\ker(A)$	Nullity of the matrix A	
$\Lambda(A)=\{\lambda_1,...,\lambda_n\}$	Spectrum of the matrix A	
$A\otimes B$	Kronecker product	87
$\mathrm{supp}(v)$	Support of the vector v	47
$\mathbf{A}$	Pattern (usually square)	49
$\mathbf{A}_\times$	Pattern $\mathbf{A}$ with stars along the diagonal	69
$\mathbf{A}(S\|.)$	Pattern $\mathbf{A}$ where the rows indexed by S were removed	69
$FO(c)$	Forward-overlap of the column c	113
$\mathscr{P}$	Dendrogram	118
$\mathscr{H}_r(\mathscr{P})$-matrix	Hierarchical matrix	119

Graphs and graph invariants

$G=(V,E)$	Graph with node set V and edge set E	19
K_n	Complete graph on n nodes	24
P_n	Path on n nodes	24
C_n $(n\neq 2)$	Cycle on n nodes	25
$B=(V^+,V^-,E)$	Bipartite graph	27

Contents

Chapter 1

Introduction

Graph theory began in 1736 with the Swiss mathematician Leohnard Euler and his paper on the problem known under the name *The seven bridges of Königsberg* [33]. The city of Königsberg (now, Kaliningrad, located in Russia) was divided into four lands by the Pregel River. This river was spanned by seven bridges, each of them connecting one land to another. It is said that the people of Königsberg asked themselves if there was a walk through the city crossing each bridge exactly once and possibly returning to its starting point. The answer was brought by Euler who proved that it was impossible to find such a route. In [33], Euler provided a necessary and sufficient condition for the existence of a walk of the above type in an arbitrary graph.

One century after Euler's solution to the seven bridge problem, the British mathematician Arthur Cayley studied a class of graphs that he called *trees* [17]. Thereafter, Cayley applied his results on trees to chemistry, which became the first application branch of graph theory [18]. It is only in 1878 in a paper by the British mathematician James Joseph Sylvester about graphs and molecular diagrams that the term *graph* was introduced [92].

Nowadays, graphs are of application in many different areas of engineering and science, including biology, physics, chemistry and even sociology, where they are used to represent problems in an abstract fashion.

Although graphs are easily defined, their numerous applications gave rise to a series of non trivial mathematical problems, amongst which is the famous *four color problem* introduced in 1852 by the South African mathematician Francis Guthrie. The problem was formulated as follows: given a map in the plane, is it possible to color all the countries with four colors in such a way that two adjacent countries are colored differently ? For more than one century, several mistaken proofs were suggested, including one by Cayley, and the problem remained unsolved. Finally, in 1969, the German mathematician Heinrich Heesch proposed the first proof based on computers [43].

The *maximum flow problem* was another famous concern of graph theory. This problem was first stated in 1954 by the American mathematician Theodore Edward Harris whose purpose, in collaboration with the General F.S. Ross, was to apply the results of this problem to the Soviet railway network [88]. The problem consists in finding a flow in a given graph, starting from a particular node called the *source* and going to another particular node called the *sink*, that is maximum. In 1956, the American mathematicians Lester Randolph Ford junior and Delbert Ray Fulkerson proved the well-known *max-flow min-cut theorem* stating that the maximum flow is equal to the minimum capacity that, when removed from the graph in a specific way, prevents any flow from going from the source to the sink. The max-flow min-cut theorem was applied in different topics of graph theory [34, 37], but also in matrix theory. Several results about the existence of matrices satisfying combinatorial constraints were indeed proved by means of this theorem [14]. This combination of graph and matrix theory with combinatorics led to the combinatorial matrix theory.

Combinatorial matrix theory, defined as the branch of mathematics combining graph theory, combinatorics and linear algebra, includes among others the combinatorial proofs of classical algebraic theorems, such as the Cayley-Hamilton theorem, and the study of matrices with prescribed combinatorial properties [12, 14].

This thesis is concerned with four different topics of combinatorial matrix theory: the minimum rank problems, the structural controllability of networked systems, the binary factorizations of the all ones matrix and finally, nearly-linear time solvers for linear systems. Except for linear solvers, the motivation for us to work on these topics was brought by the finite-time average consensus problem. In the framework of the average consensus, we are given a number of communicating agents which all have an initial position and which are driven by a dynamics of the following form: at each time step, each agent receives a limited number of positions and with the received information, it changes its own position following a given law. The problem consists in finding a communication graph with a law to change the position of the agents so that after a finite number of steps, all the agents meet at the average of their initial positions. Mathematically, the problem is finding a weighted graph for which there is a power of the adjacency matrix that is the matrix of all ones divided by the number of agents.

Our first idea to tackle this problem was to study graphs for which the powers of the adjacency matrix have their ranks decreasing to one. This led us to the minimum rank problems, the first topic of the thesis. Besides our contri-

butions to the minimum ranks, we applied the theory developed for minimum ranks to structural controllability, the second topic we consider in the sequel.

Still motivated by the average consensus problem, we studied the binary matrices, seen as the adjacency matrices of graphs, that are matrix roots of the matrix of all ones. More specifically, we compared them with the De Bruijn graphs which are known to be an optimal strategy to reach the consensus as quickly as possible. This is the third topic of the thesis.

Finally, independently of the average consensus, we chose to work on nearly-linear time solvers for linear systems, one of the attractive research topics of the last ten years.

In the rest of this introduction, we introduce each of these topics separately.

Minimum rank problems

Graphs and matrices are related by the correspondence that can be made between the position of the edges in the graph and the position of the nonzero entries in the matrix. A matrix is said to be underlying a given graph if its nonzero entries match the position of the edges in the graph, meaning that the (i, j)-entry is nonzero if and only if there is an edge from j to i in the graph. The minimum rank problems consist in finding the minimum possible rank for a matrix underlying a given graph.

The minimum rank problems were initiated in 1996 by Nylen [79] when the graph is simple and undirected. By simple, we mean that the graph has no self-loop and that the matrices underlying the graph have a diagonal that is non-constrained. Due to the symmetry of the undirected graphs, the matrices underlying such graphs are naturally required to be symmetric.

Nylen's motivation for the minimum rank problem of a simple undirected graph was the Inverse Eigenvalue Problem of a Graph (IEPG) [35, 46]. The distinction between the IEPG and spectral graph theory is worth to be mentioned. Spectral graph theory aims to study the spectrum of particular matrices underlying a simple undirected graph, such as the adjacency or the Laplacian matrix, in order to get information about the graph. As an example, it has been shown [46] that the eigenvalues of the adjacency matrix give the number of edges in the graph. Instead, the goal of the IEPG is to use the characteristics of the simple undirected graph in order to deduce all the possible spectra for the matrices underlying the graph.

Although the IEPG has been studied for more than 20 years, limited progress has been made. However, as Nylen had noticed, a first step forward in this problem would be to determine the maximum possible multiplicity of an eigenvalue

for a matrix underlying a simple undirected graph. This maximum multiplicity is actually provided by the minimum rank of the graph [79]. That is how the minimum rank problems were introduced.

Since originally stated for simple undirected graphs, most of the literature on the minimum rank problems concerns this type of graphs, e.g. [6, 11, 35, 57, 58, 79]. Nevertheless, these five last years have seen some breakthroughs in the determination of the minimum rank of other types of graphs, some of which are directed or with self-loops, e.g. [5, 47].

The progress in the problem of computing the minimum rank of a graph was made through several graph invariants, ranging from the path cover number [49] and the zero forcing number [3,5,47] to the minimum degree [11] and the clique cover number [35]. The zero forcing number was introduced specifically for the minimum rank problems, at first in [3] for simple undirected graphs and then in [5,47] for the other types of graphs. This invariant was introduced in order to solve the minimum rank problem for each kind of trees [47]: this problem was indeed reduced to the one of computing the zero forcing number. However, the author of [1] proved the NP-hardness of computing the zero forcing number of a simple undirected graph.

Chapter 3 first gives an overview of the minimum rank problem when the graph is simple and undirected, then presents the graph invariant called zero forcing number before providing our contributions on zero forcing and minimum rank. Our first result completes the one of [1] by proving that the non-equivalent problem of computing the zero forcing number in directed graphs allowing self-loops is also NP-hard. Then, our second contribution shows that the minimum rank of some directed trees can be computed in linear time.

Our result on NP-hardness was published in [94]:

M. Trefois and J.-C. Delvenne, *Zero forcing number, constrained matchings and strong structural controllability*, Linear Algebra and Its Applications, 484: 199-218, 2015.

Structural controllability

Dynamical networked systems, or *networked systems* for short, are omnipresent in our daily life. The internet, aircrafts, robots, power grids or electrical circuits are only a few examples. As their name suggests, networked systems are related to graphs. A networked system should indeed be thought of as a graph with a dynamical process on it.

Our ability to control networked systems reflects our understanding of how they work. The notion of controllable system matches our intuitive idea of what controllability is. A networked system is called *controllable* if it can be driven in finite time from any initial state to any desired final state. Specifically, a controllable networked system can be led by an outside controller which directly acts on a part of the nodes in the underlying graph; these nodes form the input set. This notion of control gives thus rise to two questions, classical in control theory: is a given networked system controllable ? If so, what is the minimum size of an input set ?

The controllability of networked systems was studied from a graph theoretic approach in [93] when the dynamical process is driven by the Laplacian matrix of the underlying graph. Further progress was then made in [31, 83, 105] when the Laplacian matrix leads the dynamical process and in [39] when it is instead the adjacency matrix which drives the dynamics. For instance, it has been shown in [83] that controllability can be deduced from the eigenvectors of the Laplacian matrix. Other results about the controllability of networked systems using graph theoretic arguments can be found in [16,39,70,75,83,93,103,105].

Most of the developments about the controllability of networked systems assume the interaction strengths, or equivalently the weights along the edges of the underlying graph, to be known. However, in many real networked systems, such as the regulatory networks, the weights along the edges are unknown or only partially determined. As a consequence, the well-known conditions for controllability, such as the *Kalman controllability rank condition* [54], are not applicable for such systems. That is why another variant of controllability, called *structural controllability*, has been introduced.

Structural controllability takes into account only the topology of the underlying graph, but not the interaction strengths along the edges. We distinguish two kinds of structural controllability: the weak one introduced in 1974 by Lin [66] and the strong one defined by Mayeda and Yamada in 1979 [71].

A networked system is called *weakly* structurally controllable, or *weakly controllable* for short, if there exist interaction strengths making the system controllable. In contrast, a system is called *strongly* structurally controllable, or *strongly controllable* for short, if the system is controllable whatever the interaction strengths.

The last four years have seen a surge of activity in structural controllability [19, 22, 67, 77]. The results of [19] and [67] link structural controllability with maximum matchings. More specifically, in [67], Liu, Slotine and Barabási

re-prove a result of [21] that states that weak controllability can be deduced from the maximum matchings in the underlying graph whereas Chapman and Mesbahi prove in [19] the link between strong controllability and maximum constrained matchings. The authors of [77] make use of the zero forcing sets in the underlying graph, supposed to be simple and directed, in order to deduce the strong controllability of the system. The zero forcing sets are related to the zero forcing number, which, as said previously, was introduced in the framework of the minimum rank problems.

Chapter 4 emphasizes the role of zero forcing in controllability. Precisely, our contributions revisit through zero forcing sets the results of [19] about the relation between strong controllability and constrained matchings. Firstly, we provide a necessary and sufficient condition in terms of zero forcing sets for the strong controllability of networked systems from a given input set. Secondly, in the case of self-damped systems, i.e. systems in which each node influences itself, we show that minimum-size input sets for strong controllability are provided by the minimum zero forcing sets in the simple graph that is the underlying graph where self-loops were all removed. Finally, we show that we can find in polynomial time a minimum-size input set for the strong controllability of a self-damped system with a tree structure. Following [19, 22, 67, 80], the results of this chapter are for directed graphs allowing self-loops.

All of these results were published in [94]:

M. Trefois and J.-C. Delvenne, *Zero forcing number, constrained matchings and strong structural controllability*, Linear Algebra and Its Applications, 484: 199-218, 2015.

Binary factorizations of the all ones matrix

Graphs with a unique walk of given length m from any source node to any target node are of interest in different mathematical problems. In [73] for example, it is shown that these graphs are used in the construction of a class of algebras. Or, in the framework of the finite time average consensus problem stated in [29], such graphs are among the communication topologies reaching the consensus the most quickly.

The adjacency matrices of these graphs are actually the binary solutions to the equation $A^m = J$, where J is the matrix of all ones. Nowadays, these solutions are far from being well understood: the only known solutions are the De Bruijn matrices which are the adjacency matrices of the De Bruijn graphs, introduced in [26].

Most of the literature [59,68,100,101] about the binary solutions to $A^m = J$ considers the solutions that are g-circulant, meaning that each row (except the first one) is obtained from the previous one by shifting all the entries of g positions to the right. For example, the authors of [101] showed that some of the g-circulant solutions to $A^m = J$ are isomorphic to a De Bruijn matrix.

Even though the De Bruijn matrices are the only known solutions to $A^m = J$, the binary solutions to this equation have been proved [69] to share several properties with the De Bruijn matrices. The authors of [69] even call them *Generalized De Bruijn matrices*. As a consequence, you may wonder if each binary solution to $A^m = J$ can be expressed from De Bruijn matrices.

Chapter 5 improves our understanding of the binary solutions to $A^m = J$ and, in particular, investigates the relation between these solutions and the De Bruijn matrices. As a first result, we show that each binary solution to $A^m = J$ with minimum rank is isomorphic to a row permutation of a De Bruijn matrix and that this row permutation can be represented by a block diagonal permutation matrix. From this result, we deduce a characterization of the minimum rank solutions to $A^2 = J$ through the De Bruijn matrices. This partially solves Hoffman's open problem of characterizing the binary solutions to $A^2 = J$. Finally, we identify a class of roots A of J that are isomorphic to a De Bruijn matrix.

These results were published in [96]:

M. Trefois, P. Van Dooren and J.-C. Delvenne, *Binary factorizations of the matrix of all ones*, Linear Algebra and Its Applications, 468: 63-79, 2015.

Linear solvers with fast iterations

The problem of solving very large linear systems arises in different contexts. For instance, in many areas of physics, mechanics and electro-magnetics, the solutions of partial differential equations have to be determined numerically and a spatial discretization of the problem naturally leads to the one of solving a large sparse linear system [104].

In order to solve a linear system, there exist two main approaches. The first one makes use of *direct methods* [25], such as the Cholesky factorization or the Gauss elimination, that provide the exact solution of the system after a finite number of computations, but that might be computationally expensive. The second one uses *iterative methods* [32, 85, 104], such as the Jacobi method or the gradient descent, that are generally less costly in terms of running time, but

that only give an approximation of the system solution. However, when the size n of the system is large, both direct and iterative methods have a prohibitive computational complexity. For example, the Gauss elimination method runs in $\mathcal{O}(n^3)$ time and the Jacobi method in $\mathcal{O}(Nn^2)$ time where N is the number of needed iterations. During the last decade, the hope has been to find an iterative algorithm with a nearly-linear running time. In this context, 'nearly-linear' means a running time that is of the form $\mathcal{O}(m \log^c m)$, where m is the size of the system or the number of nonzero entries in the system matrix, and c is an arbitrary constant. This problem has been considered for symmetric and diagonally dominant (SDD) systems. In 2004, Spielman and Teng showed in [90] that algorithms for SDD systems with nearly-linear running time are indeed possible. Fuelled by these seminal results, there has been a surge of interest in developing nearly-linear time iterative methods for solving an SDD system. Since 2004, several algorithms have been proposed [20, 56, 60–62, 65, 90]. All of these reduce the problem of solving an SDD system to the one of solving a Laplacian system of the form $Lx = b$, where L is the Laplacian matrix of a simple undirected graph and efficiently solve it in nearly-linear time by using graph theoretic techniques.

Recently, a nearly-linear time direct method that solves systems where the coefficient matrix is a special case of hierarchical matrix has been proposed in [4]. Hierarchical matrices were introduced in 1999 by Hackbusch [42] in order to perform efficiently matrix operations involving dense matrices. Originally, their introduction comes from partial differential equations (PDE's). Solving PDE's by finite element methods, for instance, often gives dense matrices or sparse matrices with dense inverse. However, scientists try to avoid dense matrices because they make matrix operations expensive. In order to deal efficiently with them, the so-called wavelet approaches were introduced [24]. The idea is to provide sparse matrices from special functions in discretization. Hierarchical matrices extend this idea and are used as data sparse approximations of dense matrices. Moreover, as it is shown in [41], arithmetic operations on hierarchical matrices, such as matrix-vector multiplication or matrix inversion, can be performed in nearly-linear time. Hierarchical matrices are then computationally powerful. That is why nowadays they are also used to construct preconditioners in order to speed up the resolution of linear systems [102].

In this chapter, we contribute to finding nearly-linear time solvers. Indeed, we consider the problem of computing the minimal norm solution of an underdetermined linear system and provide a new iterative algorithm that beats all the current iterative methods thanks to its running time per iteration that is

faster than linear in the system size. More specifically, our algorithm iterates on the columns q of a matrix that we call k-sparsely factorizable and performs in $\mathcal{O}(k)$ time the iteration

$$x_{t+1} = x_t + \frac{x_t^T q}{q^T q} \cdot q. \tag{1.1}$$

In particular, we show that if the k-sparsely factorized matrix is hierarchical, the running time per iteration depends only on some characteristics of the hierarchical structure, such as the depth or the degree, but not directly on the matrix size. Thanks to this and some existing results on low-stretch spanning trees [2,56], we deduce that we can find in nearly-linear time the minimal norm solution of a linear system where the coefficient matrix is the incidence matrix of a graph. Finally, we show that our algorithm can be used to solve a Laplacian system in nearly-linear time. The idea is to factorize the Laplacian matrix through the incidence matrix, then to change the variable and to find from our algorithm the minimal norm solution to a system where the coefficient matrix is the incidence matrix, before finally coming back to the original variable by solving in linear time an almost triangular system.

These results were submitted as a journal paper in 2016 [95]:

M. Trefois, M.T. Schaub, J.-C. Delvenne and P. Van Dooren, *A new sparse matrix factorization for linear solvers with fast iterations*, submitted, 2016.

The present introduction is followed by an overview of the basic concepts of graph theory we widely use throughout this thesis, see Chapter 2. The next four chapters are about the above mentioned topics and have been written to be read independently: Chapter 3 is about the *minimum rank problems*, Chapter 4 about *structural controllability*, Chapter 5 about the *binary factorizations of the all ones matrix* and Chapter 6 about *linear solvers with fast iterations*. Finally, Chapter 7 concludes our investigations about these four topics.

Chapter 2

Preliminaries

This chapter is an introdution to *graph theory*, widely used throughout this book. At first, we present the notion of graph, as well as different matrix representations. Then, the notion of a matching in a graph is presented.

2.1 Graph and matrix representation

In real life, graphs are everywhere: people exchanging messages, computers physically connected by cables, metropolitan networks connecting cities are a few examples among others.

A graph is a mathematical tool allowing to represent interacting objects in an abstract fashion.

In graph theory, the objects are called *nodes* and the interactions between the objects are called *edges*.

2.1.1 Definition of graph

Mathematically, a graph is a pair of *finite* sets: the *non-empty* set of nodes and the set of edges. An edge is a pair $\{i, j\}$ of two nodes, modelling the fact that information can flow between nodes i and j.

Definition 2.1. *A graph, or an undirected graph, is a pair* (V, E) *of finite sets where* V *is non-empty and* E *contains pairs of elements in* V*, namely* $E \subseteq \{\{i, j\} : i, j \in V\}$*. The set* V *is the set of nodes and* E *is the set of edges.*

The edges connecting a node to itself, namely the edges of the form $\{i, i\}$, are allowed and called *self-loops*, or *loops* for short.

A natural way to represent a graph is by drawing it. Graphically, a graph is represented by a set of items, modelling the nodes, which are linked by lines, modelling the edges. If $\{i, j\}$ is an edge of the graph, then we draw a line

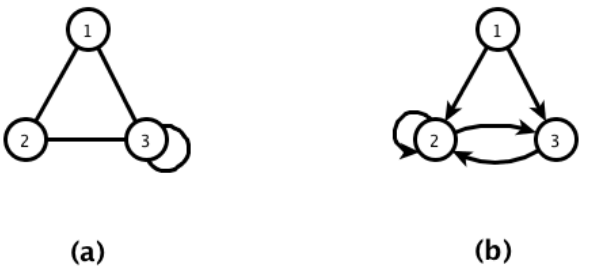


Figure 2.1: (a) A graphical representation of the graph (V,E) with $V = \{1,2,3\}$ and $E = \{\{1,2\},\{1,3\},\{2,3\},\{3,3\}\}$. - (b) A graphical representation of the *directed* graph (V,E) with $V = \{1,2,3\}$ and $E = \{(1,2),(1,3),(2,2),(2,3),(3,2)\}$.

between the items representing nodes i and j. As an example, Figure 2.1 (a) is a graphical representation of the graph (V,E) with $V = \{1,2,3\}$ and $E = \{\{1,2\},\{1,3\},\{2,3\},\{3,3\}\}$.

The number of nodes in a graph G is called the *order* of G and is denoted by $|G|$.

In a given graph, nodes i and j are called *neighbors* if $\{i,j\}$ is an edge of the graph. In particular, node i is a neighbor of itself if the loop $\{i,i\}$ is in the graph. The number of neighbors of node i is called the *degree* of i and is denoted by $\deg(i)$. For instance, in the graph shown in Figure 2.1, the neighbors of node 3 are nodes 1, 2 and 3 and $\deg(3) = 3$.

Nodes i and j are called the *endpoints* of the edge $\{i,j\}$ and $\{i,j\}$ is said to be *incident* to nodes i and j. The degree of a node i can then be seen as the number of edges incident to i.

Given a graph $G = (V,E)$ and a node subset $S \subseteq V$, $G-S$ denotes the graph obtained from G by removing the nodes in S and the edges of G incident to a node of S. If $S = \{i\}$ contains a unique node i, $G-S$ is also denoted by $G-i$. As an example, in Figure 2.2, we show a graph G and the graph $G-\{4,5\}$ obtained from G by removing nodes 4 and 5.

We have seen that an edge $\{i,j\}$ models the fact that information can flow between nodes i and j. However, there are systems where information can go from i to j, but not from j to i. To model them, we use *directed* graphs.

Definition 2.2. *A directed graph is a pair (V,E) of finite sets where*

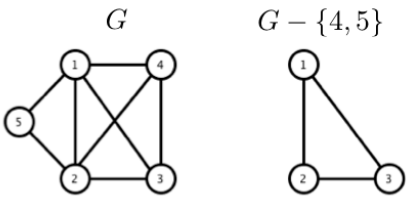


Figure 2.2: A graph G and the graph $G - \{4, 5\}$ obtained from G by removing nodes 4 and 5.

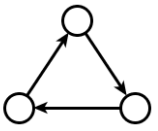


Figure 2.3: An example of 1-regular directed graph.

$E \subseteq V \times V := \{(i, j) : i, j \in V\}$. *The non-empty set V is the node set and E is the edge set.*

As for undirected graphs, a directed graph is easily represented by drawing. Graphically, it is represented by a set of items, modelling the nodes, which are linked by arrows, modelling the edges. If (i, j) is an edge of the graph, then we draw an arrow from the item representing nodes i to the one representing node j. As an example, Figure 2.1 (b) shows a graphical representation of the directed graph (V, E) with $V = \{1, 2, 3\}$ and $E = \{(1, 2), (1, 3), (2, 2), (2, 3), (3, 2)\}$.

In a given directed graph, node j is called an *out-neighbor* of node i if (i, j) is an edge of the graph. In this context, we also say that node i is an *in-neighbor* of node j.

Given an edge $e = (i, j)$, node i is called the *source node* of edge e and node j is the *target node* of e. In a directed graph G, the *out-degree* of node i, denoted by $\deg_{out}(i)$, is the number of edges in G for which the source node is node i. Similarly, the *in-degree* of node i, denoted by $\deg_{in}(i)$, is the number of edges in G for which the target node is node i. In Figure 2.1 (a), the out-degree of node 3 is 1 whereas its in-degree is 2.

A directed graph is said to be *p-regular* if each node has out and in-degree p. Figure 2.3 shows a 1-regular directed graph.

We say that two undirected (resp. directed) graphs (V, E) and (V', E') are

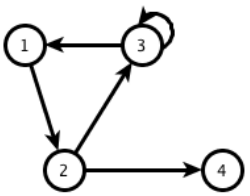


Figure 2.4: The sequence $(1,2,3,1,2,4)$ is a walk of length 5 from node 1 to node 4.

isomorphic if there is a bijection f between the node sets V and V' such that for each pair of nodes $i,j \in V$, $\{i,j\} \in E$ (resp. $(i,j) \in E$) if and only if $\{f(i),f(j)\} \in E'$ (resp. $(f(i),f(j)) \in E'$).

Walking in a graph is one of the basic tools of graph theory.

Definition 2.3. *In a directed graph $G=(V,E)$, a walk of length l from node i to node j is a sequence $i_1,\ldots,i_{l+1}$ of $l+1$ not necessarily distinct nodes such that $i_1=i$, $i_{l+1}=j$, for each $1 \le s \le l$, $(i_s,i_{s+1}) \in E$.*

As an example, in the graph shown in Figure 2.4, the sequence $(1,2,3,1,2,4)$ is a walk of length 5 from node 1 to node 4.

A *subgraph* of an undirected (resp. directed) graph $G=(V,E)$ is an undirected (resp. directed) graph $G'=(V',E')$ where the node set of G' is a subset of the node set of G, i.e. $V' \subseteq V$, and the edge set of G' is a subset of the edge set of G, i.e. $E' \subseteq E$.

Definition 2.4. *Let $G=(V,E)$ be a graph and $V' \subseteq V$ be a node subset of V. The subgraph of G induced by V' is the graph where the node set is V' and where the edges are all the edges of G for which the endpoints are in V'. When V' is clear by context, such a subgraph is called node-induced.*

In a graph, a node k is said to be *connected* to edge $\{i,j\}$ if $i=k$ or $j=k$ or both.

Definition 2.5. *Let $G=(V,E)$ be a graph and $E' \subseteq E$ be an edge subset of E. The subgraph of G induced by E' is the graph where the edge set is E' and where the nodes are the ones of G connected to an edge of E'. When E' is clear by context, we call such a subgraph edge-induced.*

Now, imagine we are given a directed graph modelling a community of people exchanging messages, namely each node corresponds to a person and if

person i sends messages to person j, the couple (i, j) is an edge of the directed graph. Given any two pairs of people, does the graph tell us if a pair exchanges much more messages than the other ? Unfortunately, not. In order to remediate this problem, we could add a positive number on each edge in such a way that the bigger this number is, the more intensive the interaction is. This leads to the notion of *weighted graph*.

Definition 2.6. *A weighted undirected (resp. directed) graph is a triplet* (V, E, W) *where* (V, E) *is an undirected (resp. directed) graph and* W *is the weight function defined by:*

$$W : V \times V \to \mathbb{R} : (i, j) \mapsto \begin{cases} w_{ij} > 0 & \text{if } \{i, j\} \in E \ (resp.\ (i, j) \in E) \\ 0 & \text{otherwise.} \end{cases}$$

An undirected (resp. directed) graph should be thought of as a weighted undirected (resp. directed) graph where the weight function takes value 1 on each edge.

If the weights along the edges are different from 1, a graphical representation of a weighted graph contains the value of the weights on the lines/arrows representing the edges.

When the graph is weighted, the degree or the in- and out-degrees of a node take into account the value of the weights along the edges, namely in a weighted undirected graph (V, E, W), the degree of node i is given by:

$$\deg(i) := \sum_{j \in V} W(i, j).$$

Similarly, in a weighted directed graph (V, E, W), the out-degree of node i is given by:

$$\deg_{out}(i) := \sum_{j \in V} W(i, j)$$

and the in-degree of i is:

$$\deg_{in}(i) := \sum_{j \in V} W(j, i).$$

A subgraph of a weighted graph $G = (V, E, W)$ is a directed graph $G' = (V', E', W')$ where the graph (V', E') is a subgraph of (V, E) and W' is the weight function W restricted to $V' \times V'$.

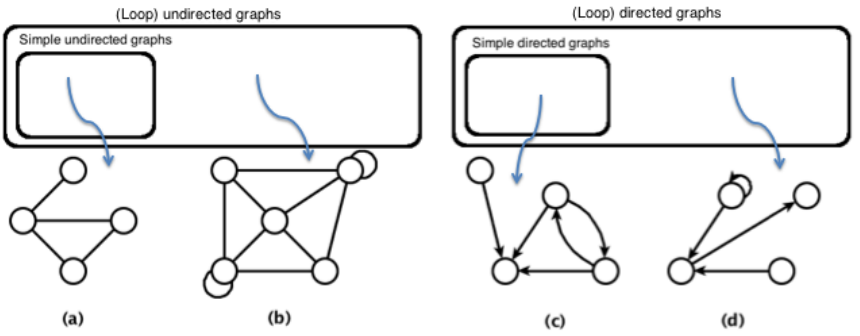


Figure 2.5: (a)-(c) An example of graph that is both a loop and a simple (un)directed graph. - (b)-(d) The given graph is a loop (un)directed graph, but not a simple (un)directed graph.

Throughout the thesis, we always specify if the graph under consideration is directed or not. Moreover, when the graphs are required to have no loop, we refer to them using the expression *simple* (un)directed graphs. Instead, when the graphs are allowed to have loops on their nodes, we refer to them by the term *loop* (un)directed graphs. In other words, the class of the loop (un)directed graphs is the whole class of (un)directed graphs whereas the simple (un)directed graphs are included in the loop ones. When no confusion can be made, we simply call a graph with potential loops on its nodes *(un)directed graph* instead of *loop (un)directed graph*. A summary of the different types of graphs is given in Figure 2.5.

2.1.2 Particular graphs

In what follows, we introduce some particular graphs, namely the complete graphs, the paths, the cycles, the trees, the directed trees and the bipartite graphs.

Definition 2.7. *The complete graph K_n on n nodes is, up to isomorphism, the simple undirected graph (V,E) where the node set is $V = \{1,...,n\}$ and that contains all the possible edges, namely the edge set is $E = \{\{i,j\} : 1 \leq i < j \leq n\}$.*

Definition 2.8. *The path P_n on n nodes is, up to isomorphism, the simple undirected graph (V,E) where the node set is $V = \{1,...,n\}$ and where the edge set is $E = \{\{i,i+1\} : 1 \leq i \leq n-1\}$.*

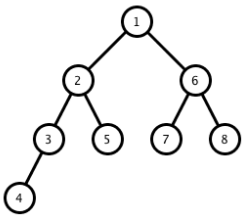


Figure 2.6: An example of tree.

The path P_n on n nodes has $n-1$ edges and is said to be of *length* $n-1$. The two nodes of degree 1 in P_n ($n \geq 2$) are called the *endpoints* of the path.

In an undirected graph, a path between nodes i and j is a subgraph isomorphic to a path P_n where the endpoints are i and j. A *shortest path* between nodes i and j is a path between i and j of minimum length.

In an undirected graph, the paths tell us how well the nodes are connected.

Definition 2.9. *An undirected graph is connected if there is a path between each pair of nodes. Otherwise, the graph is disconnected.*

Definition 2.10. *The connected components of an undirected graph G are the subgraphs of G induced by a node subset that are connected and maximal, meaning that the addition of a node to the induced subgraph makes it disconnected.*

Definition 2.11. *The cycle C_n on n nodes ($n \neq 2$) is, up to isomorphism, the undirected graph (V, E) where the node set is $V = \{1, ..., n\}$ and where the edge set is $E = \{\{i, i+1\} : 1 \leq i \leq n-1\} \cup \{1, n\}$.*

Note that the cycle C_1 is a node with a loop.

An undirected graph is said to be *acyclic* or *without cycle* if it has no subgraph which is a cycle. The undirected graphs that are connected and without cycle are called *trees*.

Definition 2.12. *A tree is a connected undirected graph without cycle. A forest is a disjoint union of trees.*

Typically, a tree is a graph like the one in Figure 2.6.

Proposition 2.13. *A tree T on n nodes has exactly $n-1$ edges.*

In a tree, a node of degree 1 is called a *leaf*.

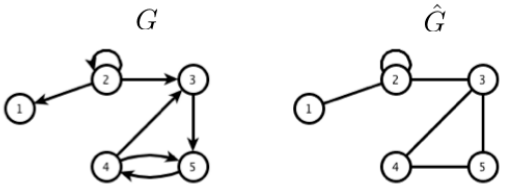


Figure 2.7: A loop directed graph G with its associated undirected graph $\hat{G}$.

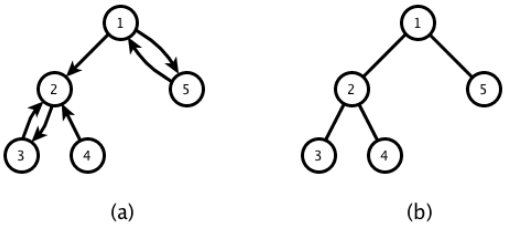


Figure 2.8: (a) A directed tree T. - (b) The undirected graph $\hat{T}$ associated with the directed tree T in (a).

Proposition 2.14. *Between two given nodes of a tree, there is a unique path.*

The unique path between two nodes i and j of a tree is denoted by $P(i, j)$.

A *rooted tree* is a tree where a node has been designated to be the root. Given a node i of a rooted tree, a *child* of i is a node j connected to node i, namely edge $\{i, j\}$ is in the tree, such that the unique path $P(r, j)$ between the root r and node j is made up of the unique path $P(r, i)$ between r and i and edge $\{i, j\}$. Node i is then called the *father* of node j. For instance, in the tree shown in Figure 2.6, if node 1 is the root, then node 4 is a child of node 3.

As their name suggests, *directed trees* are directed graphs with a tree structure. Below, we give a formal definition.

Definition 2.15. *The undirected graph $\hat{G}$ associated with a loop directed graph G has the same node set as G and for any nodes i, j, the pair $\{i, j\}$ is an edge of $\hat{G}$ when at least one of $(i, j), (j, i)$ is an edge of G.*

In Figure 2.7, we give a loop directed G and its associated undirected graph $\hat{G}$.

Definition 2.16. *A directed tree is a directed graph for which the associated undirected graph is a tree.*

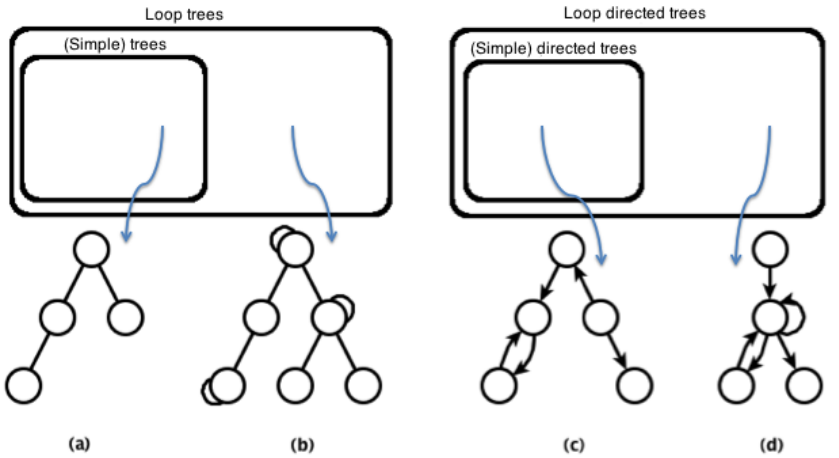


Figure 2.9: (a)-(c) Example of simple (directed) trees, which are also loop (directed) trees. - (b)-(d) Loop (directed) trees that are not simple (directed) trees.

An example of directed tree T is shown in Figure 2.8 (a) and its associated undirected graph $\hat{T}$ is shown in Figure 2.8 (b). We notice that $\hat{T}$ is a tree.

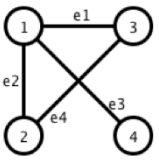
It is important to note that between two nodes i and j, a directed tree allows the two edges (i, j) and (j, i).

By definition, the trees or directed trees have no loop. However, we use the term *simple (directed) tree* to refer to a (directed) tree, which by definition has no loop, and the term *loop (directed) tree* to refer to a graph that is a (directed) tree except that it allows loops on its nodes. A summary of the different types of trees is given in Figure 2.9.

In order to avoid ambiguity, we always specify if we consider the class of simple (directed) trees or the class of loop (directed) trees. When there is no ambiguity, we simply call a simple (directed) tree by *(directed) tree*.

An undirected graph where the node set can be partitioned into two sets V^+ and V^- in such a way that each edge connects a node of V^+ with a node of V^- is called a *bipartite* graph.

Definition 2.17. *A bipartite graph (V^+, V^-, E) is an undirected graph where the node set is the disjoint union $V^+ \cup V^-$ and where the edge set E is such that each edge is of the form $\{i, j\}$ with $i \in V^+$ and $j \in V^-$.*



$$A = \begin{pmatrix} & 1 & 1 & 1 \\ 1 & & 1 & \\ 1 & 1 & & \\ 1 & & & \end{pmatrix} \quad L = \begin{pmatrix} 3 & -1 & -1 & -1 \\ -1 & 2 & -1 & \\ -1 & -1 & 2 & \\ -1 & & & 1 \end{pmatrix} \quad B = \begin{array}{c|cccc} & e_1 & e_2 & e_3 & e_4 \\ 1 & 1 & -1 & 1 & \\ 2 & & 1 & & 1 \\ 3 & -1 & & & -1 \\ 4 & & & -1 & \end{array}$$

Figure 2.10: For the given simple undirected graph, we provide its adjacency matrix A, its Laplacian matrix L, as well as an incidence matrix B.

2.1.3 Matrix representation

Although drawing a weighted graph provides a nice visualization of it, this kind of representation may not be handy. For this reason, we present some *matrix representations* which allow us to represent a weighted graph by means of a matrix.

Given an (un)directed graph on n nodes and m edges, since two different labelings of the nodes and edges provide isomorphic graphs, we suppose without loss of generality that the nodes are labeled from 1 to n and the edges are labeled from 1 to m.

The following matrices, depending on the node and edge numbering, are unique up to a row and column permutation.

Each of the following matrix representations is illustrated in Figure 2.10.

A first matrix representing a weighted graph is the *adjacency matrix*.

Definition 2.18. *Let $G = (V, E, W)$ be a weighted directed (resp. undirected) graph on n nodes. The adjacency matrix $A = (a_{ij})_{n\times n}$ of G is the $n \times n$ matrix defined as*[*]*:*

$$a_{ij} = \begin{cases} w_{ji} & \text{if } (j,i) \in E \ \ (\text{resp. } \{i,j\} \in E) \\ 0 & \text{otherwise.} \end{cases}$$

[*]In graph theory, the adjacency matrix is often defined as the transpose of this matrix. This choice for the definition of the adjacency matrix is to be consistent with the control literature (see Chapter 4).

Although the adjacency matrix is very convenient, the Laplacian matrix of a simple undirected graph, which offers another representation of the graph, is also of interest.

Definition 2.19. *Let $G = (V, E, W)$ be a simple and weighted undirected graph on n nodes. The Laplacian matrix L of G is the $n \times n$ matrix defined by*

$$L = D - A,$$

where $A \in \mathbb{R}^{n\times n}$ is the adjacency matrix of G and $D \in \mathbb{R}^{n\times n}$ is the diagonal matrix of the degrees, namely the i^{th} diagonal entry of D is the degree of node i.

We refer the reader to [74] for an interesting survey about Laplacian matrices.

Finally, we define an incidence matrix of a simple undirected graph. It also provides a representation of such a graph and is convenient to factorize the Laplacian matrix. Whereas the Laplacian and the adjacency matrices are node-by-node matrices, an incidence matrix is a node-by-edge matrix.

Definition 2.20. *Let $G = (V, E)$ be a simple undirected graph on n nodes and m edges. Choose an arbitrary direction on each edge of the graph, namely replace each edge $\{i, j\}$ by either edge (i, j) or edge (j, i). The incidence matrix $B = (b_{ie})_{n\times m}$ of G is then the $n \times m$ matrix defined by:*

$$b_{ie} = \begin{cases} 1 & \text{if node } i \text{ is the source node of edge } e \\ -1 & \text{if node } i \text{ is the target node of edge } e \\ 0 & \text{otherwise.} \end{cases}$$

Of course, an incidence matrix of a simple undirected graph depends on the directions chosen on the edges. However, an incidence matrix of such a graph will be used to factorize the Laplacian matrix and this factorization holds whatever the incidence matrix.

Given a weighted undirected graph $G = (V, E, W)$ on m edges, we denote by $\overline{W}$ the $m \times m$ diagonal matrix of the weights along the edges. The weight on edge $e = \{i, j\}$ is referred to as w_e or w_{ij}.

Proposition 2.21. *Given a simple and weighted undirected graph $G = (V, E, W)$, the Laplacian matrix L of G can be factorized as:*

$$L = B\overline{W}B^T,$$

where B is an incidence matrix of (V,E).

Proof. The matrix multiplication states that the (i,j)-entry of $B\overline{W}B^T$ is given by

$$\sum_{e\in E} b_{ie}b_{je}w_e.$$

We then deduce that

- if $i=j$, $\sum_{e\in E} b_{ie}b_{je}w_e = \sum_{k:\{i,k\}\in E} w_{ik} = \deg(i)$
- if $i\neq j$, $\sum_{e\in E} b_{ie}b_{je}w_e = \begin{cases} -w_{ij} & \text{if } \{i,j\}\in E \\ 0 & \text{otherwise .} \end{cases}$

Consequently, $L = B\overline{W}B^T$. □

2.2 Matching

Let us consider the following situation: a professor would like to send five master students in five different universities for an internship. Initially, he thought to spread the students randomly in the different institutions. However, his task turned out to be much more difficult due to some students' requirements for not going in some institutions. Respecting each student's constraints, our professor hopes to be able to spread the students in the different universities.

The professor's problem can be modelled with graph theory. Indeed, consider the bipartite graph (V^+,V^-,E) where V^+ is the set of nodes modelling the students and V^- is the set of nodes modelling the universities, with an edge between student i and university j if and only if student i is willing to go to university j. The problem of our professor can be expressed as: finding a set of five edges so that no two edges share a node. In terms of graph theory, our professor is actually looking for a *5-matching*.

Here is the definition of matching in a simple undirected graph.

Definition 2.22. *Let $G=(V,E)$ be a simple undirected graph on m edges and $t\in\{1,...,m\}$.*

A t-matching of G is a set of t edges such that no two edges share a node.

Given a matching, the nodes of G belonging to an edge of the matching are called matched nodes, whereas the other nodes are unmatched nodes.

A t-matching is said to be maximum in G if there is no s-matching in G with $s>t$.

Finding a maximum matching in a simple undirected graph can be done in polynomial time thanks to the following theorem due to Berge [10] about maximum matchings and augmenting paths.

Definition 2.23. *Let $\mathscr{M}$ be a matching in a simple undirected graph G.*

An $\mathscr{M}$-alternating path in G is a path in which the edges belong alternatively to $\mathscr{M}$ and not to $\mathscr{M}$.

An $\mathscr{M}$-augmenting path is an $\mathscr{M}$-alternating path where the endpoints are unmatched nodes.

Theorem 2.24. *[10] Let G be a simple undirected graph. A matching $\mathscr{M}$ of G is maximum if and only if G has no $\mathscr{M}$-augmenting path.*

From Theorem 2.24, polynomial time algorithms computing a maximum matching have been discovered, first for bipartite graphs [48] and then for general simple undirected graphs [30, 76]. If the graph has n nodes and m edges, these algorithms run in time $\mathscr{O}(\sqrt{n}m)$.

However, as shown in [87], computing a maximum matching in a simple tree is an easier task.

Theorem 2.25. *[87] In a simple tree, a maximum matching can be found in linear time.*

The notion of *constrained* matching was originally defined in the paper of Hershkowitz and Schneider [44] for bipartite graphs.

Definition 2.26. *Let G be a simple undirected graph.*

A t-matching of G is called constrained if there is no other t-matching in G with the same matched nodes.

A constrained t-matching is said to be maximum in G if there is no constrained s-matching in G with $s > t$.

In the graph shown in Figure 2.11 (a), the 2-matching $\mathscr{M} = \{\{1,5\},\{2,4\}\}$ is constrained since there is no other 2-matching with matched nodes $1, 2, 4$ and 5. In contrast, in the graph shown in Figure 2.11 (b), the matching $\mathscr{M} = \{\{2,6\},\{3,5\}\}$ is not constrained since $\mathscr{M}' = \{\{2,5\},\{3,6\}\}$ is another 2-matching with the same matched nodes as $\mathscr{M}$.

Unlike the matchings, finding a maximum *constrained* matching in a simple undirected graph is NP-hard.

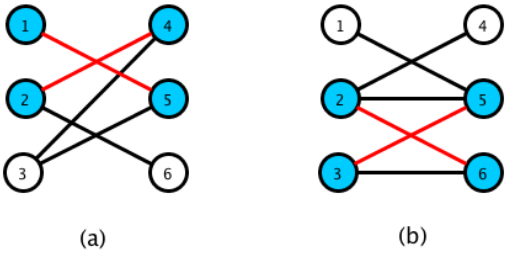


Figure 2.11: (a) The matching $\mathscr{M} = \{\{1,5\},\{2,4\}\}$ is constrained. - (b) The matching $\mathscr{M} = \{\{2,6\},\{3,5\}\}$ is not constrained.

Theorem 2.27. *[40] Finding the size of a maximum constrained matching in a bipartite graph is an NP-hard problem.*

However, it was proved in [40] that a maximum constrained matching in a simple tree can be found in linear time. This is a consequence of Theorem 2.25 and the following result.

Theorem 2.28. *[40] Each matching in a simple tree is a constrained matching.*

Corollary 2.29. *[40] In a simple tree, a maximum constrained matching can be found in linear time.*

In 2011, the notion of matching was extended to loop directed graphs by Liu, Slotine and Barabási [67].

Definition 2.30. *Let G be a loop directed graph.*

A t-matching of G is a set of t edges such that no two edges share their source nodes or share their target nodes.

A t-matching of G is maximum if there is no s-matching in G with $s > t$.

Given a matching, the nodes of G that are the target nodes of edges in the matching are called matched nodes. The other nodes are unmatched nodes.

Figure 2.12 shows examples of maximum matching in loop directed graphs.

In a loop directed graph on n nodes and m edges, a maximum matching can be found in time $\mathscr{O}(\sqrt{n}m)$ using the algorithms computing a maximum matching in a bipartite graph. Indeed, we show below how to associate a bipartite graph B_G with a loop directed graph G in such a way that computing a maximum matching in G is equivalent to computing a maximum matching in B_G.

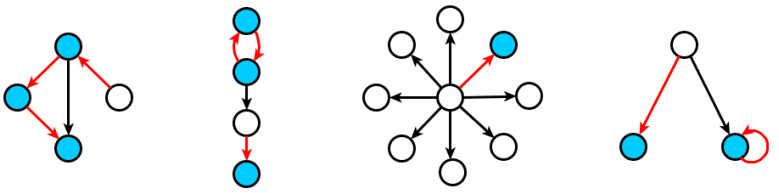


Figure 2.12: In each loop directed graph, an example of maximum matching is given. The red edges form the matching, the blue nodes are the matched nodes and the white nodes are the unmatched nodes.

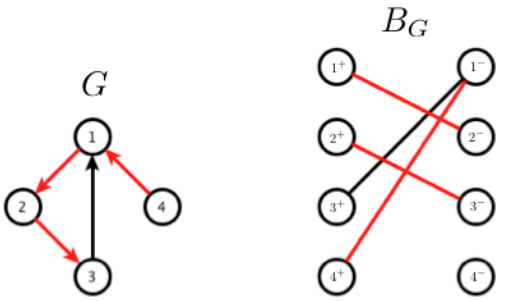


Figure 2.13: A simple undirected graph G with its associated bipartite graph B_G. The matching formed by the red edges in G corresponds to the matching with the red edges in B_G.

Definition 2.31. *Let G be a loop directed graph on nodes $i_1, ..., i_n$. The bipartite graph $B_G = (V^+, V^-, E)$ associated with G is defined as follows: the sets V^+ and V^- are two copies of the node set of G. The nodes in V^+ are denoted by $i_1^+, ..., i_n^+$ whereas the nodes in V^- are denoted by $i_1^-, ..., i_n^-$. Given any node $i_k^+ \in V^+$ and any node $i_l^- \in V^-$, $\{i_k^+, i_l^-\}$ is an edge in B_G if and only if there is an edge from node i_k to node i_l in G.*

It follows there is a one-to-one correspondence between the matchings of a loop directed graph G and the matchings of B_G.

Figure 2.13 shows a loop directed graph G with its associated bipartite graph B_G. Moreover, the red edges of G form a matching, for which the corresponding matching in B_G is highlighted in red.

Chapter 3

Minimum rank problems

The adjacency matrix of a weighted graph is determined on the one hand by the *position* of the edges in the graph and on the other hand by the *weights* along the edges.

In this chapter, given a graph G, we consider not only the adjacency matrix of G, but a *set* of real matrices, including the adjacency matrix, where the *positions* of the nonzero entries matches the *positions* of the edges in G.

The matrix set under consideration depends on the type of the graph. When the graph is undirected , the matrices are required to be symmetric and in the case of a simple graph, the diagonal is free. Here is a formal definition of the matrix family according to the type of the graph:

- if $G = (V, E)$ is a simple undirected graph,

$$\mathcal{Q}_{su}(G) := \{A \in \mathbb{R}^{|G|\times|G|} : A^T = A \text{ and for each } i \neq j, a_{ij} \neq 0 \Leftrightarrow \{i, j\} \in E\},$$

- if $G = (V, E)$ is a simple directed graph,

$$\mathcal{Q}_{sd}(G) := \{A \in \mathbb{R}^{|G|\times|G|} : \text{for each } i \neq j, a_{ij} \neq 0 \Leftrightarrow (j, i) \in E\}^*,$$

- if $G = (V, E)$ is a loop undirected graph,

$$\mathcal{Q}_{lu}(G) := \{A \in \mathbb{R}^{|G|\times|G|} : A^T = A \text{ and for each } i, j, a_{ij} \neq 0 \Leftrightarrow \{i, j\} \in E\},$$

- if $G = (V, E)$ is a loop directed graph,

$$\mathcal{Q}_{ld}(G) := \{A \in \mathbb{R}^{|G|\times|G|} : \text{for each } i, j, a_{ij} \neq 0 \Leftrightarrow (j, i) \in E\}^\dagger.$$

*In the literature of minimum rank, the matrices in $\mathcal{Q}_{sd}(G)$ are defined as the transpose of these matrices. As for the adjacency matrix, we chose to transpose the matrices in $\mathcal{Q}_{sd}(G)$ in order to be consistent with the control literature in Chapter 4.

†Same remark as for the matrices in $\mathcal{Q}_{sd}(G)$.

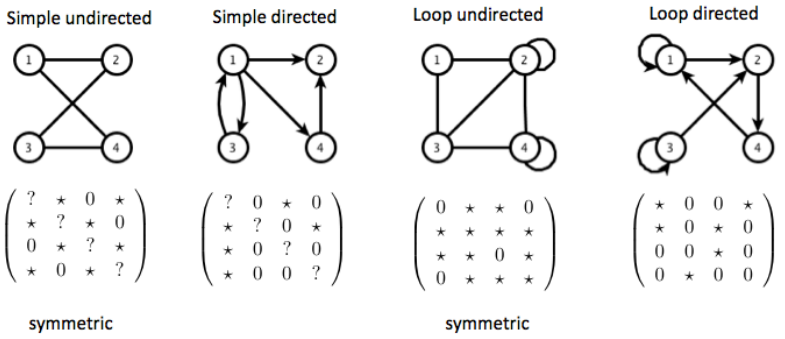


Figure 3.1: For each graph, the pattern of the matrices in its matrix set is given. A star $\star$ is used to denote a nonzero entry and a question mark ? is used to denote an entry that can be zero or nonzero. It is specified when the matrices are required to be symmetric.

We call *pattern* a matrix where the entries are either a star $\star$, or a question mark ?, or zero. A star $\star$ denotes a nonzero entry and a question mark ? denotes an entry that can be zero or nonzero. In Figure 3.1, we provide a summary of these four matrix families. For each case, we give an example of graph with the pattern of the matrices in its matrix set.

The minimum rank of a graph G considered as a graph of type i, with $i =$ su (simple undirected), sd (simple directed), lu (loop undirected) or ld (loop directed), is the minimum *possible* rank for a matrix in $\mathcal{Q}_i(G)$, namely

$$\mathrm{mr}_i(G) := \min\{\mathrm{rank}(A) : A \in \mathcal{Q}_i(G)\}.$$

Since throughout this chapter the type i of the graph G is always clearly stated, we make an abuse of notation and denote the minimum rank by $\mathrm{mr}(G)$ instead of $\mathrm{mr}_i(G)$.

The issue of computing the minimum rank of a graph is called the *minimum rank problem*.

The minimum rank problem was originally stated for simple undirected graphs. As a matter of fact, most of the literature about minimum rank problems is devoted to simple undirected graphs, e.g. [6, 11, 35, 57, 58, 79]. Section 3.1 is an introduction to the *standard* minimum rank problem, namely the minimum rank problem when the graph is simple and undirected.

The zero forcing number is a graph invariant introduced in order to study

the minimum rank of the graph, whatever its type. Section 3.2 introduces the zero forcing number and shows the link between the zero forcing number and the minimum rank of the graph. Moreover, we prove our first result stating that the computation of the zero forcing number in a loop directed graph is NP-hard. This completes Aazami's similar result [1] for simple graphs.

In the literature, most of the progress [3, 5, 27, 35, 47, 49–52, 79] in the minimum rank problems is about simple or loop (directed) trees. However, although there are several efficient algorithms for the minimum rank of a simple or loop tree, polynomial time algorithms, if they exist, for the minimum rank of simple or loop *directed* trees are still missing. Section 3.3 first presents the existing algorithms computing the minimum rank of simple or loop trees and then proves our second result that the minimum rank of particular loop directed trees can be computed in linear time.

Section 3.4 contains concluding remarks for this chapter.

3.1 Standard minimum rank problem

This section is an introduction to the *standard* minimum rank problem, namely the minimum rank problem when the graph is simple and undirected. Given such a graph $G = (V, E)$, recall that the matrix set under consideration is

$$\mathcal{Q}_{su}(G) := \{A \in \mathbb{R}^{|G|\times|G|} : A^T = A \text{ and for each } i \neq j, a_{ij} \neq 0 \Leftrightarrow \{i, j\} \in E\}.$$

The problem consists in computing the minimum rank $\mathrm{mr}(G)$ of G, namely the minimum possible rank for a matrix in $\mathcal{Q}_{su}(G)$.

Computing the minimum rank of a simple undirected graph, and even of any type of graph, is a challenging problem. However, it is worth mentioning the following straightforward properties about the minimum rank of a simple undirected graph G:

1) If the connected components of G are $G_1, ..., G_t$, then $\mathrm{mr}(G) = \sum_{i=1}^{t} \mathrm{mr}(G_i)$.

2) If G' is a node-induced subgraph of G, then $\mathrm{mr}(G') \leq \mathrm{mr}(G)$.

3) $\mathrm{mr}(G) \leq |G| - 1$.

   Indeed, a matrix $A \in \mathcal{Q}_{su}(G)$ with a rank that does not exceed $|G| - 1$ can be obtained from any matrix A' of $\mathcal{Q}_{su}(G)$ with eigenvalue λ by taking $A := A' - \lambda I$.

4) If G is connected, $\mathrm{mr}(G) = 1$ if and only if $G = K_n$.

   It is indeed straightforward that $\mathrm{mr}(K_n) = 1$. Moreover, since G is connected, each matrix of $\mathcal{Q}_{su}(G)$ has no zero row and no zero column. In addition, a matrix of rank 1 with no zero row and no zero column has only nonzero entries. It then follows that if $\mathrm{mr}(G) = 1$, then G must be the complete graph.

5) If G is a path, then $\mathrm{mr}(G) = |G| - 1$.

   Indeed, if we remove the first row and the last column from any matrix of $\mathcal{Q}_{su}(G)$, we are left with an upper triangular matrix where the diagonal has no zero entry. Therefore, $\mathrm{mr}(G) \geq |G| - 1$. Finally, it follows from the third property that $\mathrm{mr}(G) = |G| - 1$.

It is also interesting to mention that the paths are the only simple undirected graphs G for which the minimum rank is $|G| - 1$.

Theorem 3.1. *[36] If the simple undirected graph G is such that $\mathrm{mr}(G) = |G| - 1$, then G is a path.*

Corollary 3.2. *If G is a cycle of order at least 3, $\mathrm{mr}(G) = |G| - 2$.*

Proof. Since G is not a path, $\mathrm{mr}(G) < |G| - 1$. Moreover, G contains the path on $|G| - 1$ nodes as node-induced subgraph. Therefore, $\mathrm{mr}(G) \geq |G| - 2$. Consequently, $\mathrm{mr}(G) = |G| - 2$. □

The Inverse Eigenvalue Problem of a Graph (IEPG), intensively studied during the last twenty years, has been the main motivation for the standard minimum rank problem. The goal of Section 3.1.1 is to present the IEPG and its connection with the minimum rank.

The minimum rank of a simple undirected graph has been studied through several graph invariants. The goal of Section 3.1.2 is to present some graph invariants as well as their role in the computation of the minimum rank.

3.1.1 Inverse Eigenvalue Problem of a Graph

Spectral graph theory aims to study the spectrum of particular matrices provided by a simple undirected graph, such as the adjacency or the Laplacian matrix, in order to get information about the graph. For instance, if $\Lambda(A) =$

$\{\lambda_1, ..., \lambda_n\}$ is the spectrum of the adjacency matrix A of a simple undirected graph, then the number of edges in the graph is equal to $\frac{\sum_i \lambda_i^2}{2}$[‡].

Spectral graph theory is also used to determine whether two simple undirected graphs are isomorphic [23, 72, 89, 97], or to deduce information on the diameter of a simple undirected graph, i.e. the maximum length of a shortest path between any two nodes of the graph [14, 64].

The goal of the Inverse Eigenvalue Problem of a Graph (IEPG) is to use the characteristics of a simple undirected graph G in order to deduce all the possible spectra for the matrices in $\mathcal{Q}_{su}(G)$.

The IEPG is a difficult problem: even though it has been studied for more than twenty years, very little progress has been made. Actually, most of the results are about simple trees. In particular, the IEPG has been solved for the following particular simple trees: paths, double paths, stars, generalized stars and double generalized stars. For a definition of these simple trees and results about their inverse eigenvalue problem, we refer the reader to [7, 50, 51].

Since the IEPG is a tricky problem, a first step would be to determine the maximum possible multiplicity of an eigenvalue for a matrix in $\mathcal{Q}_{su}(G)$, or equivalently the maximum multiplicity of the graph.

The *maximum multiplicity of a simple undirected graph* G, denoted by $M(G)$, is defined as:

$$M(G) := \max\{m_A(\lambda) : A \in \mathcal{Q}_{su}(G), \lambda \in \Lambda(A)\},$$

where $m_A(\lambda)$ denotes the multiplicity of eigenvalue λ in the symmetric matrix A.

Below, we show that $M(G)$ is equal to the maximum multiplicity of any eigenvalue λ for a matrix in $\mathcal{Q}_{su}(G)$, namely: for each $\lambda \in \mathbb{R}$,

$$M(G) = M_\lambda(G) = \max\{m_A(\lambda) : A \in \mathcal{Q}_{su}(G)\}.$$

In order to prove this equality, we show that for any real numbers λ_1, λ_2, $M_{\lambda_1}(G) = M_{\lambda_2}(G)$. If a matrix $A_1 \in \mathcal{Q}_{su}(G)$ is such that $m_{A_1}(\lambda_1) = M_{\lambda_1}(G)$, then the matrix $A_2 := A_1 + (\lambda_2 - \lambda_1)I$ is such that $A_2 \in \mathcal{Q}_{su}(G)$ and $m_{A_2}(\lambda_2) = M_{\lambda_1}(G)$. Therefore, we must have $M_{\lambda_1}(G) = M_{\lambda_2}(G)$.

[‡]This property comes from the fact that the number of edges equals $\frac{\text{trace}(A^2)}{2}$ and that $\text{trace}(A^2) = \sum_{i=1}^n \lambda_i^2$.

Consequently, since for each symmetric matrix A, $m_A(0) = \dim\ker A$, we deduce that $M(G) = M_0(G) = \max\{\dim\ker A : A \in \mathcal{Q}_{su}(G)\}$ and that

$$M(G) + \mathrm{mr}(G) = |G|.$$

Therefore, computing $M(G)$ is equivalent to compute $\mathrm{mr}(G)$. This is how the standard minimum rank problem was introduced.

For a survey about the inverse eigenvalue problem and spectral graph theory, we refer the reader to [46].

3.1.2 Graph invariants

The minimum rank of a simple undirected graph has been studied through several graph invariants. Whereas some of them, such as the clique cover number or the minimum degree, are well known invariants, others, such as the path cover number or $\Delta(G)$, were defined in order to compute the minimum rank.

The clique cover number provides an upper bound on the minimum rank.

Definition 3.3. *Let G be a simple undirected graph.*

- *A subgraph G' of G is a clique if G' is a complete graph.*
- *A clique covering of G is a set of subgraphs which are cliques and such that each edge of G is contained in at least one of these cliques.*
- *The clique cover number of G, denoted by $cc(G)$, is the minimum number of cliques in a clique covering of G.*

It is well-known [35] that the clique cover number of a simple undirected graph is an upper bound on the minimum rank.

Proposition 3.4. *If G is a simple undirected graph,* $\mathrm{mr}(G) \leq cc(G)$.

The minimum degree of a node has been shown [11] to provide an upper bound on the minimum rank of *some* simple undirected graphs, but it has also been conjectured [11, 13] this upper bound should hold for *each* simple undirected graph.

The minimum degree of a node in an undirected graph G is denoted by $\delta(G)$.

From Properties 4) and 5) and Corollary 3.2, it follows that if G of order at least two is a complete graph, or a path, or a cycle, then $\mathrm{mr}(G) = |G| - \delta(G)$.

Moreover, from Property 3), it is straightforward that if G is a simple undirected graph with $\delta(G) \leq 1$, then $\mathrm{mr}(G) \leq |G| - \delta(G)$.

The following relation between the minimum rank of a bipartite graph G and $\delta(G)$ has been proved in [11].

Theorem 3.5. *[11] If G is a bipartite graph,* $\mathrm{mr}(G) \leq |G| - \delta(G)$.

In [11], it has also been proved that the inequality $\mathrm{mr}(G) \leq |G| - \delta(G)$ holds under some conditions on the minimum degree $\delta(G)$.

Proposition 3.6. *[11] Let G be a simple undirected graph. If $\delta(G) \leq 3$ or $\delta(G) \geq |G| - 2$, then* $\mathrm{mr}(G) \leq |G| - \delta(G)$.

Recall that the difference between a simple *undirected* graph and a simple *directed* graph that is *symmetric* is that in the latter case the matrices under consideration for the minimum rank are not required to be symmetric.

Theorem 3.7. *[11] If G is a simple directed graph that is symmetric,* $\mathrm{mr}(G) \leq |G| - \delta(G)$.

However, although for a simple directed graph G that is symmetric, $|G| - \delta(G)$ is an upper bound on the minimum rank, this result has not yet been proved in the case of a simple undirected graph.

Conjecture 3.8. *[11, 13] If G is a simple undirected graph,* $\mathrm{mr}(G) \leq |G| - \delta(G)$.

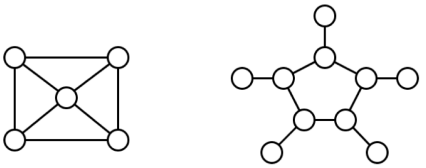
The two following invariants have been defined in [49] in order to solve the minimum rank problem in the case of a simple tree.

Definition 3.9. *[49] For a simple undirected graph G,*

$$\Delta(G) := \max\{p - q : \textit{ there is a set of } q \textit{ nodes whose deletion leaves } p \textit{ paths}\}.$$

This invariant offers an upper bound on the minimum rank. Indeed, it was shown in [49] that for each simple undirected graph G, $\mathrm{mr}(G) \leq |G| - \Delta(G)$, or equivalently $\Delta(G) \leq M(G)$.

However, this invariant $\Delta(G)$ is mainly of interest in the case of a simple tree T where $\mathrm{mr}(T) = |T| - \Delta(T)$ (see Theorem 3.11).



W_5 H_5

Figure 3.2: The wheel W_5 on 5 nodes and the penta-sun H_5.

In a simple undirected graph, two paths are said to be *node-disjoint* if they do not have a common node. Moreover, a set of paths is said to be node-disjoint if any two paths of this set is node-disjoint.

A path covering in a simple undirected graph is a set of paths such that each node of the graph belongs to at least one path of this set.

Definition 3.10. *The path cover number of a simple undirected graph G, denoted by $P(G)$, is the minimum number of paths in a node-disjoint path covering of G where each path is a node-induced subgraph of G.*

In [49], it has been shown the minimum rank of a simple tree T can be expressed in terms of $\Delta(T)$ or $P(T)$.

Theorem 3.11. *[49] For each simple tree T, $M(T) = \Delta(T) = P(T)$, or equivalently,* $\mathrm{mr}(T) = |T| - P(T) = |T| - \Delta(T)$.

Thanks to the previous theorem, several algorithms computing the minimum rank of a simple tree have been deduced. Some of them are described in Section 3.3.

For a simple undirected graph G however, $M(G)$ and $P(G)$ are not comparable. Indeed, at first consider the wheel graph W_5 in Figure 3.2. By inspection, the path cover number $P(W_5)$ of W_5 equals 2. Moreover, from the following theorem proved in [9], we deduce that $\mathrm{mr}(W_5) = 2$ and therefore, $M(W_5) = 3$.

Theorem 3.12. *[9] A connected simple undirected graph G has* $\mathrm{mr}(G) \leq 2$ *if and only if G does not contain as a node-induced subgraph any of P_4, Dart, $\ltimes$, or $K_{3,3,3}$, all shown in Figure 3.3.*

Consequently, this is an example of simple undirected graph G where $M(G) > P(G)$. Now, consider the penta-sun graph H_5 shown in Figure 3.2. By inspection, we compute that the path cover number $P(H_5)$ must be equal

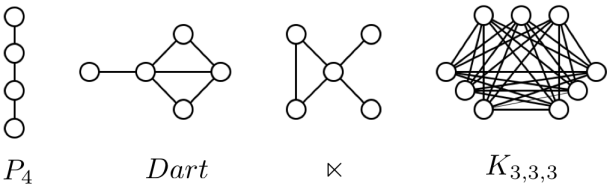


Figure 3.3: The forbidden subgraphs for a minimum rank ≤ 2.

to 3. In addition, it has been proved in [6] that $\text{mr}(H_5) = 8$, or equivalently $M(H_5) = 2$. Consequently, there also exist simple undirected graphs G for which $M(G) < P(G)$.

For more details about the relation between $P(G)$, $\Delta(G)$ and $M(G)$, we refer the reader to [6, 8].

The minimum rank problem was also extended to loop undirected graphs and simple or loop directed graphs. Since the maximum multiplicity $M(G)$ of a simple undirected graph G is equal to the maximum nullity $M_0(G)$ of a matrix in $\mathscr{Q}_{su}(G)$, we generalize the definition of $M(G)$ to each type of graph as follows: for a graph G of type i, with $i =$ su (simple undirected), lu (loop undirected), sd (simple directed), or ld (loop directed), the *maximum nullity* $M(G)$ of G is defined as:

$$M(G) := \max\{\dim\ker A : A \in \mathscr{Q}_i(G)\}.$$

Consequently, we still have $M(G) + \text{mr}(G) = |G|$.

In [5,47], the authors suggest another definition for the path cover number, but this time the definition holds for each type of graph (directed or not, with potential loops or not). The purpose was to have the following property true: for each simple or loop (directed) tree T, $M(T) = P(T)$.

In Definition 2.8, we defined a path as an undirected graph. We extend now the notion of path to a directed graph. Intuitively, a directed path is a directed graph where we can go from a node to another without possibility to come back. Figure 3.4 gives an example of such a path and below, we give a formal definition. When it is clear by context that the path is directed, we simply call it *path*.

Definition 3.13. *A directed graph on n nodes is said to be a path if it is isomorphic to the directed graph (V, E) where the node set is $V = \{1, ..., n\}$ and the edge set is $E = \{(i, i+1) : 1 \leq i \leq n-1\}$.*

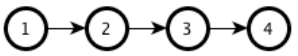


Figure 3.4: An example of directed path.

A graph G of any type is said to *require nonsingularity* if $M(G) = 0$. Otherwise, if $M(G) > 0$, we say G *allows singularity*. Note that each simple undirected graph G allows singularity since $\mathrm{mr}(G) \leq |G| - 1$.

Definition 3.14. *[5,47] The path cover number of a graph G of any type, denoted by $P(G)$, is the minimum number of node-disjoint paths whose deletion from G leaves a graph that requires nonsingularity (or is the empty set).*

It should be noticed that unlike Definition 3.10, the paths in the above definition are not required to be induced subgraphs of G.

Theorem 3.15. *[5, 47, 49] For each simple or loop (directed) tree T, $M(T) = P(T)$.*

For a general simple undirected graph, Definitions 3.10 and 3.14 give different values for the path cover number. Nevertheless, the path cover number obtained with not necessarily induced paths (Definition 3.14) remains incomparable with $M(G)$. The graphs W_5 and H_5 remain good examples. For loop (un)directed graphs G, the relation between $P(G)$ and $M(G)$ is less clear. In [5], it was asked if there exists a loop (un)directed graph G such that $M(G) < P(G)$.

In the next section, we present another graph invariant, called the zero forcing number, defined for each type of graph G and denoted by $Z(G)$, also with the property that for each simple or loop (directed) tree T, $M(T) = Z(T)$. However, unlike the path cover number, the zero forcing number is an upper bound for the maximum nullity of any graph.

3.2 Zero Forcing Number

The zero forcing number (ZFN) is a graph invariant that was initially defined for simple undirected graphs [3] in order to get a lower bound on the minimum rank of the graph. Then, the notion of zero forcing number has been extended to loop directed graphs in [5] and to simple directed graphs and loop undirected graphs in [47].

Although the zero forcing number $Z(G)$ of a graph G is an upper bound on the maximum nullity of the graph, whatever its type, the interest in this invariant lies also in the fact that for each simple or loop (directed) tree T, $M(T) = Z(T)$.

In Section 3.2.1, we give the definition of the zero forcing number for each type of graph. Then, we prove that the zero forcing number $Z(G)$ of a graph G satisfies $M(G) \leq Z(G)$. Finally, we mention that for each simple or loop (directed) tree T, $M(T) = Z(T)$.

In [1], it has been proved that computing the zero forcing number of any simple undirected graph is NP-hard. In Section 3.2.2, we prove our first result stating that the non equivalent problem of computing the zero forcing number of any loop directed graph is also NP-hard.

3.2.1 Definition

The zero forcing number has a dynamic definition. Intuitively, we start with an initial coloring of the nodes with some of them black and the others white, then we use a color change rule that will color in black some of the white nodes. The goal is to determine the minimum number of nodes that have to be initially black in order to have the whole graph black after the color change rule. This is the zero forcing number.

Below, we give a formal definition of the color change rule and of the zero forcing number. The color change rule is slightly different if the graph is simple or with potential loops on its nodes. These different rules are needed in order that whatever the type of a (directed) tree, the zero forcing number provides the exact value of the minimum rank of the tree.

The *color change rule* in a *loop* undirected (resp. directed) graph G is the following [5,47]: suppose each node of G is either black or white. Given a node i, if exactly one (out-)neighbor j of i is white (possibly $j = i$), then change the color of node j to black.

As an example, consider the loop undirected graph in Figure 3.5 (a). Since node 1 has only node 2 as white neighbor, the color change rule applied to node 1 would change the color of node 2 to black.

In a loop graph, node i needs not be black to change the color of one of its (out-)neighbors. Instead, in a simple graph, node i must be black to be able to change the color of one of its (out-)neighbors. Here is the definition of the color change rule in a simple graph.

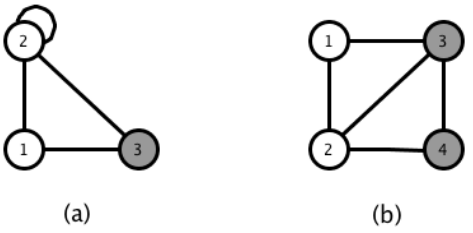


Figure 3.5: (a) A loop undirected graph. The color change rule applied to node 1 changes the color of node 2 to black. - (b) A simple undirected graph. The color change rule applied to node 4 changes the color of node 2 to black.

In a *simple* undirected (resp. directed) graph G, the *color change rule* is defined as follows [3, 47]: suppose each node of G to be black or white. Given a node i, if i is *black* and node j is the only white (out-)neighbor of i, then change the color of j to black.

For instance, consider the simple undirected graph in Figure 3.5 (b). Since node 4 is black and node 2 is the unique white neighbor of node 4, the color change rule applied to node 4 would change the color of node 2 to black.

Note that if we apply the color change rule repeatedly on the graph until no more color change is possible, then the final coloring we get is unique and does not depend on the order in which we applied the color change rule on the nodes.

Given an initial coloring, there may be different ways to color the whole graph in black using the above color change rule. However, the only thing that matters is if we can get the whole graph black by applying repeatedly the color change rule, whatever the order in which we apply it on the nodes.

Definition 3.16. *Let $G = (V, E)$ be a graph of any type.*

- *The zero forcing number $Z(G)$ of G is the minimum number of nodes which have to be initially black so that after applying repeatedly the color change rule to G all the nodes of G are black.*

- *A node subset $S \subseteq V$ with the property that if only the nodes of S are initially black in G, then the whole graph is black after applying repeatedly the color change rule is called a zero forcing set of G.*

- *A zero forcing set of size $Z(G)$ is called a minimum zero forcing set.*

The idea behind the notion of *zero forcing* is the following: given a graph G of type i (su, sd, lu, or ld) with nodes numbered from 1 to n, if a vector $v \in \mathbb{R}^n$ has zeros in the entries corresponding to the black nodes of G, changing a node from white to black when applying the color change rule means the corresponding entry in v is forced to be zero if v is in the kernel of the transpose of a matrix in $\mathcal{Q}_i(G)$. This is shown below in the proof that for each graph G, $M(G) \leq Z(G)$.

The *support* of a vector $v = (v_1, ..., v_n) \in \mathbb{R}^n$, denoted by $\text{supp}(v)$, is the set of indices $i \in \{1, ..., n\}$ such that $v_i \neq 0$.

Proposition 3.17. *[3] Let $A \in \mathbb{R}^{n\times n}$ with $\dim\ker A > k$. Then, there is a nonzero vector $v \in \ker A$ vanishing at any k specified positions. In other words, if S is a set of k indices, then there is a nonzero vector $v \in \ker A$ such that $\text{supp}(v) \cap S = \emptyset$.*

Proof. Let $S \subseteq \{1, ..., n\}$ be a set of k indices and let the subspace

$$V_k := \{v = (v_1, ..., v_n) \in \mathbb{R}^n : \text{ for each } i \in S, v_i = 0\}.$$

It is clear that $\dim V_k = n - k$. Therefore,

$$\dim(V_k \cap \ker A) = \dim V_k + \dim\ker A - \dim(V_k + \ker A) > n - k + k - n = 0.$$

Consequently, $V_k \cap \ker A \neq \emptyset$. □

Proposition 3.18. *[3,47] Let S be a zero forcing set of a graph $G = (V, E)$ of type i with $i =$ su (simple undirected), or lu (loop undirected), or sd (simple directed), or ld (loop directed) and $A \in \mathcal{Q}_i(G)$. If $v \in \ker(A^T)$ and $\text{supp}(v) \cap S = \emptyset$, then $v = 0$.*

Proof. If $S = V$, there is nothing to prove. Suppose then that $S \neq V$. Since S is a zero forcing set, there is a node $j \in S$ such that applying the color change rule on j changes the color of a node k (possibly $k = j$) to black. Since $(A^T v)_j = \sum_{z\in V} a_{zj} v_z$, the equation $(A^T v)_j = 0$ reduces to $a_{kj} v_k = 0$. Consequently, $v_k = 0$ and since S is a zero forcing set, $v = 0$. □

Theorem 3.19. *[3, 5, 47] For each graph G, $M(G) \leq Z(G)$, or equivalently $|G| - Z(G) \leq \text{mr}(G)$.*

Proof. Suppose $M(G) > Z(G)$ and let S be a minimum zero forcing set of G of type i with $i =$ su, lu, sd, or ld. Let $A \in \mathcal{Q}_i(G)$ such that $\dim\ker A = \dim\ker A^T > |S|$. By Proposition 3.17, there is a nonzero vector $v \in \ker A^T$ that vanishes on S. However, by Proposition 3.18, $v = 0$, which is a contradiction. □

However, if the graph is a simple or loop (directed) tree, equality holds.

Theorem 3.20. *[5, 27, 47, 49] If T is a simple or loop (directed) tree, $M(T) = Z(T)$, or equivalently* $\mathrm{mr}(T) = |T| - Z(T)$.

Remark 3.21. *Let G be a simple (resp. loop) directed graph that is symmetric and $\hat{G}$ its associated simple (resp. loop) undirected graph (cf. Definition 2.15). Since, unlike the matrices in $\mathcal{Q}_{su}(\hat{G})$ (resp. $\mathcal{Q}_{lu}(\hat{G})$), the matrices in $\mathcal{Q}_{sd}(G)$ (resp. $\mathcal{Q}_{ld}(G)$) are not asked to be symmetric, we have $\mathcal{Q}_{su}(\hat{G}) \subseteq \mathcal{Q}_{sd}(G)$ (resp. $\mathcal{Q}_{lu}(\hat{G}) \subseteq \mathcal{Q}_{ld}(G)$) and therefore,* $\mathrm{mr}(G) \leq \mathrm{mr}(\hat{G})$.

Following [47], there are simple (resp. loop) directed graphs G such that $\mathrm{mr}(G) < \mathrm{mr}(\hat{G})$.

However, in the case of a simple (resp. loop) directed tree T that is symmetric, since it follows from the definition of the color change rule that $Z(T) = Z(\hat{T})$, we deduce from Theorem 3.20 that $\mathrm{mr}(T) = \mathrm{mr}(\hat{T})$.

3.2.2 On the computation of the ZFN

In his PhD thesis [1], Aazami has proved the NP-hardness of computing the zero forcing number in a *simple undirected* graph, using the Directed Hamiltonian Cycle problem known to be NP-complete [55, 82].

A simple directed graph G on n nodes is said to be *Hamiltonian* if it contains a subgraph isomorphic to the graph (V, E) where the node set is $V = \{1, ..., n\}$ and the edge set is $E = \{(i, i+1) : 1 \leq i \leq n-1\} \cup \{(n, 1)\}$.

Theorem 3.22. *[55, 82] Deciding if a simple directed graph is Hamiltonian is NP-complete.*

Corollary 3.23. *[1] Computing the zero forcing number of a simple undirected graph is NP-hard.*

Because of the different color change rules, it should be noticed that computing the zero forcing number of a simple (un)directed graph can *not* be reduced to the computation of the zero forcing number of a loop directed graph and vice versa.

In this section, we prove our first contribution which completes the NP-hardness result of [1] about zero forcing by showing that the non-equivalent problem of computing the zero forcing number of any *loop directed* graph is also NP-hard.

We use a result of [40] stating that computing the size of a maximum constrained matching in a bipartite graph is NP-hard.

Two matrices A and B are said to be *permutation similar* if there are permutation matrices P_1, P_2 such that $A = P_1 B P_2$.

A *zero-nonzero pattern* $\mathbf{A}$ (or *pattern* for short) of size n is an $n \times n$ matrix with each entry being either a star $\star$ or zero. A star $\star$ refers to a nonzero entry.

A loop directed graph G on n nodes defines a zero-nonzero pattern $\mathbf{A}(G)$ of size n as follows: the entry a_{ij} of $\mathbf{A}(G)$ is a star $\star$ if and only if there is a directed edge from node j to node i in G. This pattern is called the *zero-nonzero pattern associated with* G.

Definition 3.24. *[5] Let* $\mathbf{A}$ *be a zero-nonzero pattern.*

- *A* t*-triangle of* $\mathbf{A}$ *is a* $t \times t$ *subpattern of* $\mathbf{A}$ *which is permutation similar to an upper triangular pattern where all the diagonal entries are nonzero.*
- *The triangle number of* $\mathbf{A}$ *is the maximum size of a triangle in* $\mathbf{A}$*.*
- *The triangle number* $\mathrm{tri}(G)$ *of a loop directed graph* G *is the triangle number of its associated zero-nonzero pattern* $\mathbf{A}(G)$*.*

Theorem 3.25. *[5] For each loop directed graph* G*,* $\mathrm{tri}(G) + Z(G) = |G|$.

Theorem 3.26 proved in [44] shows the link between the triangle number of a pattern $\mathbf{A}$ and the constrained matchings in a bipartite graph defined from $\mathbf{A}$.

A zero-nonzero pattern $\mathbf{A} = (a_{ij})_{n\times n}$ of size n defines a bipartite graph $B_{\mathbf{A}}$ where the node sets are $V^+ = \{1^+, ..., n^+\}$ and $V^- = \{1^-, ..., n^-\}$ and where $\{i^+, j^-\}$ is an edge in $B_{\mathbf{A}}$ if and only if the entry a_{ji} of $\mathbf{A}$ is a $\star$-entry. Then $B_{\mathbf{A}}$ is called the *bipartite graph associated with* $\mathbf{A}$.

Theorem 3.26. *[44] Let* $\mathbf{A}$ *be an* $n \times n$ *zero-nonzero pattern and* $B_{\mathbf{A}}$ *the bipartite graph associated with* $\mathbf{A}$*. Then the following statements are equivalent.*

- $B_{\mathbf{A}}$ *has a constrained* n*-matching*
- $\mathbf{A}$ *is permutation similar to a triangular pattern with nonzero diagonal elements.*

Here comes our first result.

Theorem 3.27. *[94] The computation of the zero forcing number of any loop directed graph is NP-hard.*

Proof. Given a bipartite graph B, observe that a constrained matching in a bipartite graph B' that is a node-induced subgraph of B is also a constrained matching in B. From this observation and Theorem 3.26, we deduce that the triangle number of a square pattern $\mathbf{A}$ equals the size of a maximum constrained matching in $B_{\mathbf{A}}$, the bipartite graph associated with $\mathbf{A}$. However, in [40] it was proved that the computation of the size of a maximum constrained matching in a bipartite graph is NP-hard. Moreover, we notice that each bipartite graph $B = (V^+, V^-, E)$ can be seen as the bipartite graph associated to a loop directed graph; in the case V^+ and V^- are of different size, some isolated nodes can be added to B without loss of generality. Therefore, computing the triangle number of a loop directed graph is also NP-hard. From this result and Theorem 3.25, we have highlighted the NP-hardness of the computation of the zero forcing number in any loop directed graph. □

The NP-hardness of the computation of the zero forcing number in any simple undirected graph implies NP-hardness for the zero forcing number in any simple directed graph. We have proved that computing the zero forcing number in any loop directed graph is also NP-hard. We also expect NP-hardness for the zero forcing number of any loop undirected graph. However, a thorough argument is still needed.

3.3 Minimum rank of trees and directed trees

In the previous section, we have seen that the zero forcing number solves the minimum rank problem of each simple or loop (directed) tree in the sense that this invariant reduces the problem of minimizing over an infinite set of real matrices to the problem of minimizing over a finite set of nodes. In [28], we can find a brute force computer program computing the zero forcing number of each simple or loop (directed) tree using the free open-source computer mathematics software system *Sage* [91]. However, due to its exponential running time such a program is not efficient for large trees. As a consequence, polynomial-time algorithms for computing the minimum rank of simple or loop (directed) trees are needed.

In this section, we describe the existing polynomial-time algorithms for the minimum rank of simple/loop trees and we prove our contribution that states that the minimum rank of some loop directed trees can be computed in linear time by a recursive elimination process.

3.3.1 The algorithms for trees

Several algorithms compute the minimum rank of simple trees T through the invariant $\Delta(T)$. Recall that for each simple tree T, $M(T) = \Delta(T)$ (Theorem 3.11), or equivalently $\mathrm{mr}(T) = |T| - \Delta(T)$. Algorithm 1, due to Johnson and Saiago [52], is one of these algorithms: at each step, the high degree nodes (cf. definition below) that have at least two neighbors that do not have a high degree are removed from the tree.

In a simple tree, a *high degree node* is a node with a degree at least three. The set of the high degree nodes of a simple tree T is denoted by $H(T)$.

In a simple tree $T = (V, E)$, the subgraph induced by a node subset $Q \subseteq V$ is denoted by $T[Q]$. In particular, $T[H(T)]$ denotes the subgraph of T induced by the high degree nodes of T.

Given a simple tree T, a subgraph T' of T and a node i belonging to both T and T', in order to avoid ambiguity, the degree of i in T is denoted by $deg_T(i)$, whereas the degree of i in T' is denoted by $deg_{T'}(i)$.

Algorithm 1:

Input: a simple tree T

Output: $\Delta(T)$

Set $Q = \emptyset$ and $T' = T$;

While $H(T') \neq \emptyset$

- remove from T' the set Q' of all the nodes $i \in H(T')$ such that $deg_{T'}(i) - deg_{T'[H(T')]}(i) \geq 2$;
- set $Q = Q \cup Q'$;

end

$\Delta(T) = p - |Q|$ where p is the number of connected components in $T - Q$;

Given a simple tree T on n nodes, a short analysis of Algorithm 1 shows the running time is $\mathcal{O}(n^2)$. A variant of this algorithm using generalized stars

can be found in [35]. This variant was also adapted in order to compute a node-disjoint path covering of minimum size in a simple tree [35].

Algorithm 2:

Input: a simple tree T

Output: a minimum zero forcing set S of T

Set $S = \emptyset$ and $T' = T$;

While $T' \neq \emptyset$

- if T' is a forest where each connected component has less than 3 nodes, then put a node of each component in S and set $T' = \emptyset$;
- otherwise consider an appropriate node i of T' in a connected component T_s of T';
  - ⋆ denote $k(\geq 2)$ the number of connected components of $T_s - i$ that are paths connected to i from an endpoint;
  - ⋆ for $k-1$ among them, put in S the endpoint that is a leaf in T_s;
  - ⋆ remove i from T' as well as the k paths connected to i from an endpoint;

end

Based on another idea, Nylen proposed in [79] a recursive formula for the minimum rank of a simple tree. This formula uses the notion of *appropriate node*.

Consider a simple tree T and a node i of T. Denote by $T_i^1, ..., T_i^t$ the connected components of $T - i$. If at least two of them are paths connected to i from an endpoint, then i is called an *appropriate node*.

Proposition 3.28. *[79] Each simple tree with at least three nodes has an appropriate node.*

Here is the recursive formula proved by Nylen.

Theorem 3.29. *[79] Let T be a simple tree on $n \geq 3$ nodes and an appropriate node i of T. If $T_i^1, ..., T_i^t$ denote the connected components of $T - i$, then*

$$\mathrm{mr}(T) = \mathrm{mr}(T_i^1) + ... + \mathrm{mr}(T_i^t) + 2.$$

In Chapter 4, we will see the ability of finding a *minimum* zero forcing *set* in a simple tree is of interest in control theory. From the existing methods computing the minimum rank of simple trees, we can deduce algorithms computing a *minimum* zero forcing *set*. Algorithm 2 (on page 46) is an example of such a method we deduced from Nylen's formula (Theorem 3.29): the algorithm iterates on the appropriate nodes and at each step, picks a node in all but one paths connected to the appropriate node by an endpoint.

Below, we give a proof of Algorithm 2.

Recall that the minimum rank of the path P_n on n nodes is $n-1$. Therefore, the zero forcing number of each path is 1. Moreover, it is straightforward that each minimum zero forcing set of a path is made up of one of its endpoints.

Theorem 3.30. *Algorithm 2 is correct.*

Proof. Denote by n the number of nodes in T. The proof is by induction on n.

- If $n = 1$ or 2, Algorithm 2 returns S containing a node of T. Therefore, S is trivially a minimum zero forcing set of T.

- If $n > 2$, Proposition 3.28 states that T has an appropriate node i. Denote by $T_i^1, ..., T_i^t$ the connected components of $T - i$. Suppose that $T_i^1, ..., T_i^l$ $(l \geq 2)$ are the paths connected to i from an endpoint. The node set S computed by Algorithm 2 is of the form

  $$S = S' \cup \left[\cup_{s=l+1}^{t} S_s\right],$$

  where S_s $(l+1 \leq s \leq k)$ is by induction a minimum zero forcing set of T_i^s and S' has been built by considering $l-1$ paths among $T_i^1, ..., T_i^l$, say for example $T_i^1, ..., T_i^{l-1}$, and for each of these paths the endpoint that is a leaf in T has been put in S'. When applying the color change rule to T with the nodes in S initially black, the $l-1$ components $T_i^1, ..., T_i^{l-1}$ will be colored in black as well as node i. Once i is black, the other components T_i^s $(l+1 \leq s \leq k)$ will be colored in black. Finally, the color change rule applied to i will color an endpoint of T_i^l in black and T_i^l will be entirely colored in black by the color change rule. So, S is a zero forcing set of T.

We have to prove now that S has a minimum size, i.e $\mathrm{mr}(T) = n - |S|$.

From Theorem 3.29, we know that

$$\mathrm{mr}(T) = 2 + \sum_{s=1}^{l} \mathrm{mr}(T_i^s) + \sum_{s=l+1}^{t} \mathrm{mr}(T_i^s).$$

Since for each $1 \leq s \leq l$, T_i^s is a path and for each $l+1 \leq s \leq t$, S_s is a minimum zero forcing set of T_i^s, we deduce

$$\mathrm{mr}(T) = 2 + \sum_{s=1}^{l} (|T_i^s| - 1) + \sum_{s=l+1}^{t} |T_i^s| - |S_s|.$$

Finally, since $\sum_{s=1}^{t} |T_i^s| = n-1$, $\mathrm{mr}(T) = n - (l - 1 + \sum_{s=l+1}^{t} |S_s|)$.

Hence, since $|S| = l - 1 + \sum_{s=l+1}^{t} |S_s|$, $\mathrm{mr}(T) = n - |S|$.

□

Finding an appropriate node in a simple tree on $n \geq 3$ nodes can be done in $\mathcal{O}(n)$ time with, e.g., a depth-first search. Here is an example of algorithm: start from a node x of degree at least 2 and for all the neighbors of x, walk to the neighbor and keep walking onto the unique unvisited node until arriving on a node that either is a leaf, or has a degree at least 2. After doing so for each neighbor of x, if at least two leaves were discovered, then x is an appropriate node and the algorithm is over. Otherwise, start this process again with a node of degree at least 2 discovered previously and walk to the nodes not yet visited.

In order to illustrate this algorithm, consider the simple tree given in Figure 3.6 and start with node $x = 1$. First, we walk to node 2 and stop on that node since it is of degree at least 2, then we walk to nodes 10 and 11 and stop on node 11 since it is a leaf. We discovered only one leaf, therefore node 1 is not an appropriate node. Start again on node $x = 2$ which we discovered to be of degree at least 2. First, we walk to nodes 3 and 4 and stop on node 4 since it is a leaf, then we walk to node 5 and stop there since it has degree 3 and finally, walk to nodes 8 and 9 and stops on node 9 which is a leaf. Since we discovered this time 2 leaves of the tree, i.e. nodes 4 and 9, node $x = 2$ is an appropriate node.

Since each node is visited at most once, the algorithm runs in time $\mathcal{O}(n)$ and Algorithm 2 has a running time in $\mathcal{O}(n^2)$.

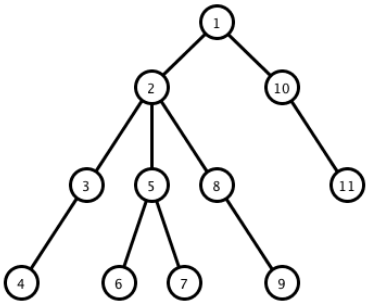


Figure 3.6: A simple tree in which node 2 is an appropriate node.

An algorithm similar to Algorithm 1 has been proposed in [27] for *loop trees*. This algorithm computes the minimum rank through a graph invariant, denoted by $\mathscr{C}_0(T)$, which is a generalization of $\Delta(T)$ to loop trees.

Let $T = (V, E)$ be a loop tree and $S \subseteq V$ be a node subset. We denote by $c_0(S)$ the number of connected components of $T - S$ that allow singularity.

Definition 3.31. *For a loop tree* $T = (V, E)$, $\mathscr{C}_0(T) := \max\{c_0(S) - |S| : S \subseteq V\}$.

The following relation between $\mathscr{C}_0(T)$ and the maximum nullity of the loop tree has been proved in [27].

Theorem 3.32. *[27] For each loop tree* T, $M(T) = \mathscr{C}_0(T)$.

Since the algorithm presented in [27] for computing $\mathscr{C}_0(T)$ is in the same idea of Algorithm 1, we leave the reader the freedom to refer to [5,27] for more details about this algorithm.

3.3.2 The algorithms for directed trees

As shown in Remark 3.21, the minimum rank of a simple (resp. loop) *symmetric* directed tree equals the minimum rank of its associated undirected graph which, by definition of a directed tree, is a simple (resp. loop) tree. As a consequence, the algorithms computing the minimum rank of simple or loop trees can be used to compute the minimum rank of simple or loop *symmetric* directed trees.

However, there is nowadays no polynomial-time algorithm computing the minimum rank of *any* simple or loop directed tree. Therefore, as stated in [47], efficient algorithms for these graphs are still needed.

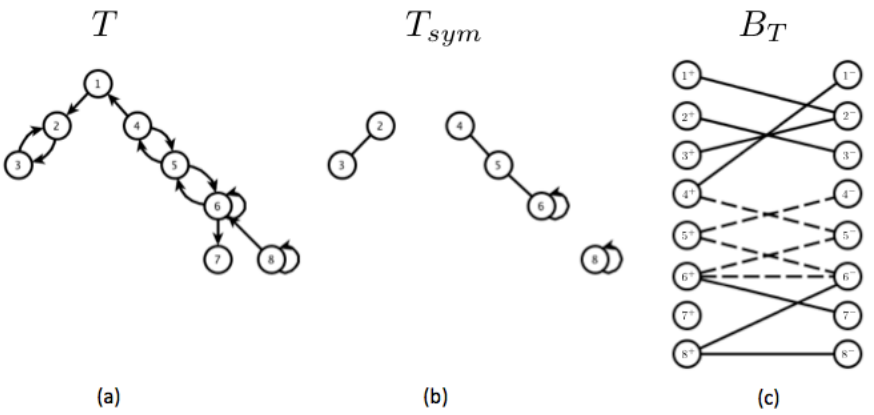


Figure 3.7: (a) A loop directed tree T with almost loop-free symmetric components - (b) The subgraph T_{sym} of T - (c) The bipartite graph B_T associated with T. The subgraph induced by the dotted edges is a path between nodes 4^+ and 4^-. This path is symmetric since its edges are $\{4^+, 5^-\}$ and $\{5^+, 4^-\}$, $\{5^+, 6^-\}$ and $\{6^+, 5^-\}$, $\{6^+, 6^-\}$.

Below, we identify a class of loop directed trees for which the minimum rank can be computed in linear time: these directed trees will be called *loop directed trees with almost loop-free symmetric components*.

In a loop directed tree T, a pair $\{i, j\}$ of nodes is called a *symmetric edge* if both edges (i, j) and (j, i) are in T. By abuse of language, we consider a symmetric edge as being an edge of T.

The subgraph T_{sym} of a loop directed tree T is the subgraph induced by the symmetric edges of T. Note that T_{sym} is a loop undirected graph.

Definition 3.33. *A loop directed tree T is said to have almost loop-free symmetric components if each connected component of T_{sym} has at most one node with a loop.*

In Figure 3.7 (a), we show an example of a loop directed tree T with almost loop-free symmetric components. Its subgraph T_{sym} is shown in Figure 3.7 (b).

The goal of what follows is to prove Theorem 3.39 stating the minimum rank of any loop directed tree with almost loop-free symmetric components can be computed in linear time.

Recall Definition 2.31 of the bipartite graph $B_G = (V^+, V^-, E)$ associated with a loop directed graph G: the sets V^+ and V^- are two copies of the node

set of G. The nodes in V^+ are denoted by $1^+, ..., n^+$ and the nodes in V^- are denoted by $1^-, ..., n^-$. Given any node $i^+ \in V^+$ and any node $j^- \in V^-$, $\{i^+, j^-\}$ is an edge in B_G if and only if there is an edge from node i to node j in G.

Definition 3.34. *Let $B = (V^+, V^-, E)$ be the bipartite graph associated with a loop directed graph. A path P in B is symmetric if for any edge $\{i^+, j^-\}$ in P, the edge $\{j^+, i^-\}$ is also in P.*

In Figure 3.7 (c), we show the bipartite graph B_T associated with the loop directed tree T in Figure 3.7 (a). The subgraph induced by the dotted edges in B_T is an example of symmetric path. Indeed, its edges are $\{4^+, 5^-\}$ and $\{5^+, 4^-\}$, $\{5^+, 6^-\}$ and $\{6^+, 5^-\}$, $\{6^+, 6^-\}$.

Following Definition 2.15, the undirected graph associated with a loop directed tree T is denoted by $\hat{T}$ and is, by definition of a loop directed tree, a loop tree. The simple tree obtained from $\hat{T}$ by removing the loops is denoted by $\hat{T}_s$.

In the bipartite graph B_T shown in Figure 3.7 (c), the path between nodes 4^+ and 4^- made up of the dotted edges is symmetric. In the following lemma, we prove that in each bipartite graph associated with a loop directed tree, each path between nodes k^+ and k^- is symmetric.

Lemma 3.35. *Let T be a loop directed tree and B_T be its associated bipartite graph. In B_T, each path between nodes k^+ and k^- is symmetric.*

Proof. Suppose there is a path P in B_T between nodes k^+ and k^- that is not symmetric, namely there is an edge $\{i^+, j^-\}$ in P with $i \neq j$ such that $\{j^+, i^-\}$ is not in P.

Since $i \neq j$, we know that $k^- \neq j^-$ and/or $k^+ \neq i^+$. Suppose without loss of generality that $k^+ \neq i^+$.

In the bipartite graph B_T, start at node k^- and go along the edges of P until edge $\{i^+, j^-\}$ is crossed. Just after crossing edge $\{i^+, j^-\}$, we arrive either at node i^+ or at node j^-.

- If we arrive at node i^+, then we deduce that P contains a path between k^- to j^- (possibly $k^- = j^-$) which contains neither edge $\{i^+, j^-\}$ nor edge $\{j^+, i^-\}$. As a consequence, we deduce that there is a path (possibly of length 0) in $\hat{T}_s$ between nodes k and j which does not contain edge $\{i, j\}$.

  Moreover, since $\{i, j\}$ is an edge of $\hat{T}_s$, we deduce that there is a path in $\hat{T}_s$ between nodes k and i *containing edge* $\{i, j\}$.

Pursuing our route along the edges of P from node i^+ to node k^+, we get a path in P between i^+ and k^+ which contains neither $\{j^+, i^-\}$ nor $\{i^+, j^-\}$. As a consequence, from this path, we deduce a path in $\hat{T}_s$ between k and i which does not contain edge $\{i, j\}$.

Consequently, we have shown in the simple tree $\hat{T}_s$, there are two different paths between nodes k and i, which contradicts Proposition 2.14 stating that in a simple tree there is a unique path between any two nodes.

- If we arrive at node j^-, we get the same contradiction with similar arguments.

□

Consider the path P between 4^+ and 4^- in Figure 3.7 (c). This path contains the nodes $4^+, 4^-, 5^+, 5^-, 6^+, 6^-$ and in T, among nodes 4, 5 and 6, node 6 has a loop. In the following lemma, we show that given any path P between nodes k^+ and k^- in B_T, at least one node i of T with i^+ and i^- in P has a loop.

Lemma 3.36. *Let T be a loop directed tree, B_T be its associated bipartite graph and P be a path in B_T between nodes k^+ and k^-. Then, walking from node k^+ to node k^- along the edges of P provides a sequence of nodes of the form either*

$$j_1^+, j_2^-, j_3^+, ..., j_{t-1}^-, j_t^+, j_t^-, j_{t-1}^+, ..., j_3^-, j_2^+, j_1^-$$

or

$$j_1^+, j_2^-, j_3^+, ..., j_{t-1}^+, j_t^-, j_t^+, j_{t-1}^-, ..., j_3^-, j_2^+, j_1^-.$$

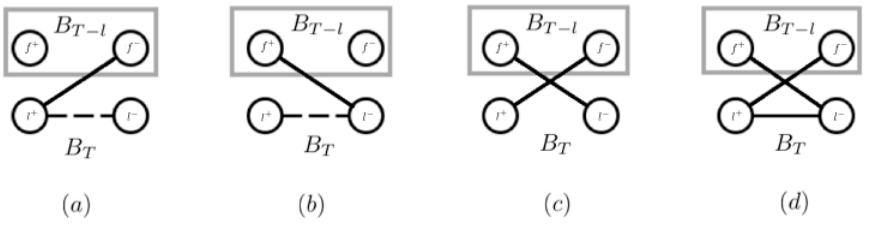
Proof. Since P is a symmetric path (Lemma 3.35), for each node i^+ of P, we know that both i^+ and i^- appear exactly once in the sequence.

In addition, due to the symmetry of P, for each node i^+ of P with $i^+ \neq k^+$, if the two neighbors of i^+ in P are i_1^- and i_2^-, then the neighbors of i^- in P are i_1^+ and i_2^+. Similarly, if the neighbor of k^+ in P is k_1^-, then the neighbor of k^- in P is k_1^+.

These observations prove the lemma by an induction starting at the endings of the sequence. □

From the previous lemma, we deduce that among the nodes i of T with i^+ (and therefore i^-) in P, at least one has a loop.

The following lemma is a direct consequence of Lemmas 3.35 and 3.36.



(a) (b) (c) (d)

Figure 3.8: In (a)-(b)-(c), if the subgraph B_{T-l} has no cycle, then B_T is acyclic as well. In (d), if B_{T-l} has a path between f^+ and f^-, then B_T has a cycle containing node l.

Lemma 3.37. *Let T be a loop directed tree, B_T be its associated bipartite graph and P be a path between nodes k^+ and k^- in B_T. Then, P is symmetric and among the nodes i of T with i^+ (and therefore i^-) in P, at least one has a loop.*

Proposition 3.38. *The bipartite graph B_T associated with a loop directed tree T having almost loop-free symmetric components is a forest.*

Proof. The proof is by induction on the number of nodes in T.

- If T has only one node, then the result is straightforward.
- Suppose by induction that the proposition holds for a tree on n nodes and prove it for a tree on $n+1$ nodes. Designate a node of T to be the root in $\hat{T}_s$. Let l be a leaf of $\hat{T}_s$ with a father f.

  We denote by B_{T-l} the bipartite graph associated with the loop directed tree $T-l$ on n nodes with almost loop-free symmetric components. By induction, we know that B_{T-l} has no cycle.

  - ⋆ If $\{l,f\}$ is not a symmetric edge of T or if $\{l,f\}$ is a symmetric edge of T and l has no loop in T, then since B_{T-l} is a forest, so it is for B_T. Indeed, this is straightforward from Figure 3.8 (a)-(b)-(c).
  - ⋆ If $\{l,f\}$ is a symmetric edge and l has a loop, there is a cycle C in B_T containing node l only if there is a path P between nodes f^+ and f^- in B_{T-l}. Indeed, this can be easily deduced from Figure 3.8 (d).

    This cycle is then made up of P, nodes l^+ and l^- and the edges $\{l^+,f^-\}$, $\{f^+,l^-\}$ and $\{l^+,l^-\}$.

    From Lemma 3.37, P is symmetric. Therefore, C is also symmetric, meaning that if $\{i^+,j^-\}$ is an edge of C, then $\{j^+,i^-\}$ is also an edge of C. More-

over, Lemma 3.37 states that at least one node k of $T - l$ with k^+ (and therefore k^-) in P has a loop.

From Lemma 3.36, we deduce that going along the edges of C starting at node l^- provides a sequence of nodes of the form either

$$l^-, f_1^+, f_2^-, f_3^+, ..., f_{t-1}^-, f_t^+, f_t^-, f_{t-1}^+, ..., f_3^-, f_2^+, f_1^-, l^+$$

or

$$l^-, f_1^+, f_2^-, f_3^+, ..., f_{t-1}^+, f_t^-, f_t^+, f_{t-1}^-, ..., f_3^-, f_2^+, f_1^-, l^+.$$

As a consequence, the nodes i of T, for which nodes i^+ and i^- are in C, are in the same connected component of T_{sym}. Therefore, T_{sym} has a connected component with at least two nodes, k and l, having a loop, which contradicts the definition of T.

□

Given a loop directed graph G on n nodes, the *zero-nonzero pattern* $\boldsymbol{A}(G)$ *associated with* G is the pattern of size n such that each entry a_{ij} is a $\star$-entry if and only if (j, i) is an edge in G.

Recall that given a pattern $\mathbf{A} = (a_{ij})_{n\times n}$, the bipartite graph $B_{\mathbf{A}} = (V^+, V^-, E)$ associated with $\mathbf{A}$ is such that $V^+ = \{1^+, ..., n^+\}$, $V^- = \{1^-, ..., n^-\}$ and $\{i^+, j^-\}$ is an edge in $B_{\mathbf{A}}$ if and only if a_{ji} is a $\star$-entry.

Note that the bipartite graph associated with a loop directed graph G is equal to the bipartite graph associated with $\mathbf{A}(G)$.

Theorem 3.39. *The minimum rank of any loop directed tree T that has almost loop-free symmetric components is computable in linear time.*

Proof. From Theorems 3.20 and 3.25, it follows that $\mathrm{mr}(T) = \mathrm{tri}(T)$. Moreover, from Theorem 3.26, we deduce that the size of a maximum constrained matching in the bipartite graph $B_{\mathbf{A}(T)}$ (or equivalently B_T) is equal to the triangle number $\mathrm{tri}(\mathbf{A}(T))$, which by definition equals $\mathrm{tri}(T)$.

Therefore, computing the minimum rank of T is equivalent to computing the size of a maximum constrained matching in B_T.

Since B_T is a forest (Proposition 3.38), Corollary 2.29, stating that computing a maximum constrained matching in a simple tree can be done in linear time, concludes the proof.

□

Below, we show it is not indispensable to compute a maximum constrained matching in B_T in order to compute the minimum rank of a loop directed tree T with almost loop-free symmetric components. Indeed, the following elimination process defined in [5] could be used.

A *realization* A of a pattern $\mathbf{A}$ is a real matrix in which an entry is nonzero if and only if the corresponding entry in $\mathbf{A}$ is a star. We write $A \in \mathbf{A}$.

The minimum rank of a pattern $\mathbf{A}$ is the minimum possible rank of one of its realizations.

Proposition 3.40. *[5] [Elimination process] Let $\mathbf{A}$ be a zero-nonzero pattern and a_{st} be a star entry of $\mathbf{A}$ such that either row s or column t or both have exactly one star entry. Then,*

$$\mathrm{mr}(\mathbf{A}) = \mathrm{mr}(\mathbf{A}_0(s|t)) + 1,$$

where $\mathbf{A}_0(s|t)$ is the pattern obtained from $\mathbf{A}$ by setting row s and column t to zero.

The minimum rank of a loop directed graph G is by definition the minimum rank of the pattern $\mathbf{A}(G)$ associated with G.

Theorem 3.41. *The minimum rank of any loop directed tree T that has almost loop-free symmetric components is computable in linear time by repeatedly applying to $\mathbf{A}(T)$ the elimination process defined in Proposition 3.40.*

Proof. When repeatedly applying the elimination process to the zero-nonzero pattern $\mathbf{A}(T)$ associated with T, if at some point the resulting pattern is not zero and is such that each nonzero row and each nonzero column has at least two $\star$-entries, this means the bipartite graph B_T contains a cycle, which is a contradiction since Proposition 3.38 states B_T is a forest. □

3.4 Conclusion

The minimum rank problems consist in computing the minimum possible rank of matrices for which the zero-nonzero pattern is determined by a given graph.

These problems are studied through several graph invariants that change the problem of minimizing over an infinite set of real matrices into the problem of minimizing over a finite set of cliques, paths, nodes, etc.

In particular, we have presented the zero forcing number $Z(G)$ of a graph G, an invariant which is of interest for the two following properties:

1. it provides a lower bound on the minimum rank, i.e. for each graph G, $|G| - Z(G) \leq \mathrm{mr}(G)$,

2. it solves the minimum rank problem of any simple or loop (directed) tree, i.e. for each such graph T, $|T| - Z(T) = \mathrm{mr}(T)$.

However, the zero forcing number is not so easy to compute. Indeed, it has been proved in [1] that computing the zero forcing number of any simple graph is NP-hard. Moreover, our first result, published in [94], completes the one of [1] and shows that computing the zero forcing number of any loop directed graph is also NP-hard. Finally, about the computation of the zero forcing number of any *loop undirected* graph, nothing has been proved yet. Even though we also expect NP-hardness for this type of graph, a thorough argument is still needed.

Most of the progress in the minimum rank problems is about simple or loop (directed) trees. However, although there exist several quadratic time algorithms computing the minimum rank of simple or loop trees, polynomial time algorithms, if they exist, for the minimum rank of directed trees are still missing. Our second result identifies a class of loop directed trees for which the minimum rank is computable in linear time using a recursive process proposed in [5].

Chapter 4

Structural controllability

A *networked system* is a linear-time-invariant system where the dynamics is driven by a matrix A provided by a graph, called the *interconnection graph* or the *underlying graph*.

Typically, a networked system is described by a differential equation of the form*:

$$\dot{x}(t) = Ax(t) + Bu(t), \tag{4.1}$$

where $A \in \mathbb{R}^{n\times n}$ is the *state matrix* which has a structure provided by a graph, $x(t) \in \mathbb{R}^n$ is the *state vector*, $B \in \mathbb{R}^{n\times m}$ $(m \leq n)$ is the *input matrix* and $u(t) \in \mathbb{R}^m$ is the *input vector*.

For instance, the matrix A could be the Laplacian or the adjacency matrix of the underlying graph.

Our ability to control such systems reflects our understanding of how these systems work. The definition of controllability matches our intuitive idea of what control is.

A system like in (4.1), denoted by the pair (A, B), is called *controllable* if it can be driven from any initial state $x_0 \in \mathbb{R}^n$ to any final state $x_f \in \mathbb{R}^n$ in finite time. In other words, given two vectors $x_0, x_f \in \mathbb{R}^n$, there is a controller $u(\cdot)$ such that starting at $x(0) = x_0$, after a finite time T the state of the system is $x(T) = x_f$.

It is well known that a system (A, B) is controllable if and only if the controllability matrix

$$C = \begin{bmatrix} B & AB & A^2B & \dots & A^{n-1}B \end{bmatrix}$$

*We point out that the dynamical systems of this chapter are in continuous time, whereas we consider discrete time systems in the following chapter.

has full rank. This is known under the name *Kalman's controllability rank condition*.

Classical controllability of a networked system was first considered in [93] when A is the Laplacian matrix of a simple undirected graph. This work then continued in [31, 83, 105] when A is the Laplacian matrix and in [39] when A is the adjacency matrix. As an example, the authors of [83] showed that controllability can be deduced from the eigenvectors of the Laplacian matrix .

More results about the controllability of networked systems using graph theoretic arguments can be found in [16, 39, 70, 75, 83, 93, 103, 105].

In a system (A, B), when a row of B is nonzero, the corresponding node in the underlying graph is said to be *directly controlled* by the outside controller. The nodes directly controlled by the controller are called the *input set*.

When A is the Laplacian matrix of a simple undirected graph, the minimum number of nodes that must be directly controlled by the outside controller in order to control the whole system has been studied for special graphs in [78, 81, 83, 105]. Computing the minimum size of an input set allowing to control the system is indeed an important issue of control theory.

Although the literature about the classical controllability of networked systems assumes the weights along the edges of the underlying graph to be completely known, in many real networked systems the weights along the edges are unknown or only partially determined. In such a case, the classical Kalman condition of controllability becomes difficult to use and the existing results about classical controllability are not applicable. That is why *structural controllability* has been introduced.

The problem of determining a control strategy for a network of interconnected systems without exact knowledge of the interaction strengths along the edges has seen a surge of activity these last four years [19, 22, 67, 77, 94], in particular regarding weak and strong structural controllability introduced in the 70's [66, 71].

Structural controllability takes into account only the structure of the interconnection graph, but not the interaction strengths along the edges.

A system with a given interconnection graph is weakly structurally controllable, or *weakly controllable* for short, from an input set S if we can choose interaction strengths making the system controllable from S.

Instead, a system with a given interconnection graph is strongly structurally controllable, or *strongly controllable* for short, from an input set S if whatever the interaction strengths, the system is controllable from S.

This chapter is about the study of the weak and strong controllability of a networked system. In particular, we also study the strong structural controllability of systems with a tree structure, used in social science for example. Indeed, sociologists and economics often try to make predictions on the behavior of people from a community organized following a given graph [38, 86]. Some of these communities, such as families or business employees, are hierarchically organized and represented by trees. In [84], the authors study the minimum number of people in a tree-organized community that have to be controlled in order to make all the people converge to a same desired behavior in finite time. In this chapter, we continue among others their work on the control of systems with a tree-structure.

Most of the papers about the classical controllability of networked systems suppose the underlying graph to be undirected. However, as said in [67], undirected graphs appear in very few applications. As a consequence, following [19, 22, 67, 80, 94], throughout this chapter the underlying graph is supposed to be a *loop directed graph*.

Section 4.1 is a statement of the weak and strong controllability problem.

Section 4.2 presents the main results of [19,21,67] linking weak and strong controllability with the notion of matching.

Section 4.3 presents our results linking strong controllability and zero forcing sets. Firstly, we show that testing if a system is strongly controllable from an input set S is equivalent to checking if S is a zero forcing set in the interconnection graph. We will see that this first result provides an intuitive quadratic-time algorithm checking if a system is strongly S-controllable. Secondly, in the case of systems that are self-damped (i.e. the state of each node is influenced, among others, by itself), we show that minimum-size input sets for strong controllability are provided by the minimum zero forcing sets in the *simple* interconnection graph, which is the interconnection graph without its loops. In particular, we show that we can find a minimum-size input set in polynomial time for the strong controllability of a self-damped system with a tree structure.

Section 4.4 shows how our results from the previous section differ from related results appeared in [77].

Section 4.5 concludes this chapter.

Our results for this chapter were published in [94]:

M. Trefois and J.-C. Delvenne, *Zero forcing number, constrained matchings and strong structural controllability*, Linear Algebra and Its Applications, 484: 199-218, 2015.

4.1 Structural controllability: problem statement

In this section, we define the weak and strong controllability of a system underlying a loop directed graph and we present the classical questions, regarding structural controllability, which we will consider in the rest of the chapter.

A loop directed graph $G = (V, E)$ on n nodes defines the matrix set

$$\mathscr{Q}_{ld}(G) = \{A \in \mathbb{R}^{n\times n} : \text{ for each } i, j, a_{ij} \neq 0 \Leftrightarrow (j, i) \in E\}.$$

Recall that an $n \times m$ pattern $\mathbf{A}$ is an $n \times m$ matrix in which each entry is either a star $\star$ or zero. A star $\star$ refers to a nonzero entry.

A *realization* of an $n \times m$ pattern $\mathbf{A}$ is an $n \times m$ matrix A where an entry is nonzero if and only if the corresponding entry in $\mathbf{A}$ is a star $\star$. We write $A \in \mathbf{A}$.

Given the node set $V = \{1, ..., n\}$ and a node subset $S = \{k_1, ..., k_m\} \subseteq V$, we define the $n \times m$ pattern $\mathbf{B}(S)$ as[†]

$$[\mathbf{B}(S)]_{ij} = \begin{cases} \star & \text{if } i = k_j \\ 0 & \text{otherwise.} \end{cases}$$

A networked system for which the underlying graph is a loop directed graph $G = (V, E)$ is a linear system of the form:

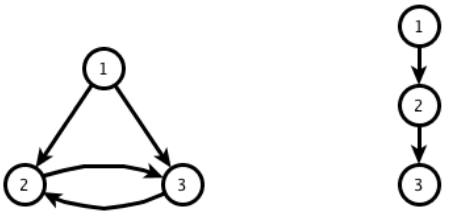
$$\dot{x}(t) = Ax(t) + Bu(t),$$

where $A \in \mathscr{Q}_{ld}(G)$, $x(t) \in \mathbb{R}^n$ is the state vector, $B \in \mathbf{B}(S)$ is a realization of a pattern $\mathbf{B}(S)$ for some input set $S \subseteq V$ and $u(t) \in \mathbb{R}^m$ is the control vector.

Notice that only the nodes in S are directly controlled by the outside controller. Such a system is referred to as system (A, B).

Given a loop directed graph G, the pattern $\mathbf{A} = \mathbf{A}(G)$ associated with G is

[†]We warn the reader that this chapter only covers systems where the outside controller independently controls each node of S. This feature appears in the structure of pattern $B(S)$ where each row and each column has at most one star entry.



(a) (b)

Figure 4.1: (a) The system underlying this loop directed graph is *weakly* S-controllable with $S = \{1\}$. - (b) The system underlying this loop directed graph is *strongly* S-controllable with $S = \{1\}$.

the pattern where the (i, j)-entry is a star $\star$ if and only if there is an edge from node j to node i in G. Therefore, each matrix of $\mathscr{Q}_{ld}(G)$ is a realization of $\mathbf{A}$.

Given an input set S, each system (A, B) where the underlying graph is G is a realization of the pair $(\mathbf{A}, \mathbf{B}(S))$, meaning that $A \in \mathbf{A}$ and $B \in \mathbf{B}(S)$. We write $(A, B) \in (\mathbf{A}, \mathbf{B}(S))$. This class of systems is referred to as system $(\mathbf{A}, \mathbf{B}(S))$.

Given a system $(\mathbf{A}, \mathbf{B}(S))$, the weak controllability seeks to know if there is a realization $(A, B) \in (\mathbf{A}, \mathbf{B}(S))$ that is controllable, in the sense that the controllability matrix has full rank.

Definition 4.1. *A system $(\boldsymbol{A}, \boldsymbol{B}(S))$ is weakly S-controllable if there is $(A, B) \in (\boldsymbol{A}, \boldsymbol{B}(S))$ that is controllable.*

Instead, the strong controllability seeks to know if each system $(A, B) \in (\mathbf{A}, \mathbf{B}(S))$ is controllable.

Definition 4.2. *A system $(\boldsymbol{A}, \boldsymbol{B}(S))$ is strongly S-controllable if all systems $(A, B) \in (\boldsymbol{A}, \boldsymbol{B}(S))$ are controllable.*

Below, we give examples of systems that are weakly or strongly controllable. These examples were given in the Supplementary Information of [67].

Consider the loop directed graph G shown in Figure 4.1 (a) with pattern $\mathbf{A}$ and the input set $S = \{1\}$. The system $(\mathbf{A}, \mathbf{B}(S))$ is described by:

$$\begin{pmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ \star_{21} & 0 & \star_{23} \\ \star_{31} & \star_{32} & 0 \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{pmatrix} + \begin{pmatrix} \star_1 \\ 0 \\ 0 \end{pmatrix} u(t).$$

The controllability matrix of this system is then:

$$C = \begin{pmatrix} \star_1 & 0 & 0 \\ 0 & \star_1\star_{21} & \star_1\star_{23}\star_{31} \\ 0 & \star_1\star_{31} & \star_1\star_{21}\star_{32} \end{pmatrix}.$$

If the nonzero values of $\star_{21}, \star_{23}, \star_{31}$ and $\star_{32}$ are such that the vectors $\begin{pmatrix} \star_{21} \\ \star_{31} \end{pmatrix}$ and $\begin{pmatrix} \star_{23}\star_{31} \\ \star_{21}\star_{32} \end{pmatrix}$ are co-linear, then the rank of C is 2. Otherwise, C has rank 3. As a consequence, since the rank of C depends on the nonzero values of $\star_{21}, \star_{23}, \star_{31}$ and $\star_{32}$, this system is weakly S-controllable.

Consider now the loop directed graph G shown in Figure 4.1 (b) with pattern $\mathbf{A}$ and the input set $S = \{1\}$. Then, the system $(\mathbf{A}, \mathbf{B}(S))$ is described by the equation:

$$\begin{pmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ \star_{21} & 0 & 0 \\ 0 & \star_{32} & 0 \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{pmatrix} + \begin{pmatrix} \star_1 \\ 0 \\ 0 \end{pmatrix} u(t).$$

The resulting controllability matrix is:

$$C = \begin{pmatrix} \star_1 & 0 & 0 \\ 0 & \star_1\star_{21} & 0 \\ 0 & 0 & \star_1\star_{21}\star_{32} \end{pmatrix}.$$

It follows that whatever the nonzero value of $\star_1, \star_{21}$ and $\star_{32}$, the matrix C has rank 3. Therefore, the system is strongly S-controllable.

In the rest of the chapter, we tackle the following questions:

Question 4.3. *Given an input set S, is the system $(\boldsymbol{A}, \boldsymbol{B}(S))$ weakly/strongly S-controllable ?*

Question 4.4. *What is the minimum size of an input set S making the system $(\boldsymbol{A}, \boldsymbol{B}(S))$ weakly/strongly S-controllable ? Can we efficiently find such a minimum-size input set ?*

Weak and strong controllability of a networked system $(\mathbf{A}, \mathbf{B}(S))$ have been characterized in [21, 67] and in [19] respectively, from the matchings in the bipartite graph B_G associated with the loop directed graph G (cf. Definition 2.31). In the next section, we present their main results.

4.2 Structural controllability and matchings

In this section, we present the main results of [21, 67] and of [19] connecting, respectively, the weak and strong controllability of a networked system with the matchings in the bipartite graph associated with the underlying loop directed graph.

First, we need to introduce the notations that will be used throughout the chapter.

In Chapter 3, a bipartite graph associated with a pattern of size n was defined. This can be extended to rectangular patterns. Let $\mathbf{A}$ be an $n \times m$ pattern. The bipartite graph $B_{\mathbf{A}}$ associated with $\mathbf{A}$ has node sets $V^+ = \{1^+, ..., m^+\}$ and $V^- = \{1^-, ..., n^-\}$. Besides, $\{i^+, j^-\}$ is an edge in $B_{\mathbf{A}}$ if and only if the (j,i)-entry of $\mathbf{A}$ is a star $\star$.

If the bipartite graph associated with a pattern $\mathbf{A}$ has a constrained t-matching, then we say by abuse of language that $\mathbf{A}$ has a constrained t-matching.

Let $S \subseteq V$ be a node subset in a loop directed graph G. A constrained S-*less* matching in the bipartite graph B_G associated with G (cf. Definition 2.31) is a constrained matching with no edges of the form $\{i^+, i^-\}$ with $i \in S$. In particular, if $S = V$, a constrained S-less matching in B_G is called a constrained *self-less* matching.

Let $\mathbf{A}$ be an $n \times m$ pattern and let $S \subseteq \{1, ..., n\}$. Then, $\mathbf{A}(S|.)$ denotes the pattern obtained from $\mathbf{A}$ by deleting the rows indexed by S.

Let $\mathbf{A}$ be a pattern of size n. Then, $\mathbf{A}_\times$ is the pattern obtained from $\mathbf{A}$ by putting stars $\star$ along the diagonal. Similarly, $G_\times$ denotes the graph obtained from the graph G by putting a loop on each node of G.

Throughout the chapter, V_{loop} is the set of nodes with a loop in the loop directed graph G underlying the system.

4.2.1 On weak controllability

In [21, 67], the authors have characterized the weak controllability of a system from the matchings in the loop directed graph underlying the system. Below, we present their result.

Recall that in a loop directed graph, a path on n nodes is a subgraph isomorphic to the directed graph (V, E) where the node set is $V = \{1, ..., n\}$ and the edge set is $E = \{(i, i+1) : 1 \leq i \leq n-1\}$.

It was proved in [21, 67] that a system with a loop directed graph G as interconnection graph is weakly controllable if and only if the controller directly acts on the unmatched nodes of a matching in G and for each matched node i, there is an unmatched node j with a path from j to i in G.

Below, we state this result with the terminology used in the rest of the chapter.

Let $G = (V, E)$ be a loop directed graph and $S \subseteq V$ a node subset of V. If for each node i of V, there is a node $j \in S$ such that there is a path in G from j to i, then the set S is called *accessible*.

Theorem 4.5. *[21, 67] Let G be a loop directed graph on n nodes with pattern $\boldsymbol{A}$ and S be an input set with cardinality $m \leq n$. System $(\boldsymbol{A}, \boldsymbol{B}(S))$ is weakly S-controllable if and only if $\boldsymbol{A}(S|.)$ has an $(n-m)-matching$ and S is accessible.*

From this theorem, it was proved in [67] that an input set of minimum size for weak controllability can be efficiently computed. This result is presented in the following theorem.

In a loop directed graph, a matching is called *perfect* if all the nodes of the graph are matched.

Theorem 4.6. *[67] Let G be a loop directed graph.*

- *If there is a perfect matching in G, then an input set of minimum size for the weak controllability of the system underlying G has size 1 and each node can be chosen to be directly controlled by the controller.*

- *If there is no perfect matching in G, the minimum size of an input set for weak controllability of the system underlying G is the number of unmatched nodes resulting from a maximum matching of G. In such a case, the unmatched nodes are an input set of minimum size.*

Since a maximum matching in a loop directed graph on n nodes and m edges can be computed in time $\mathcal{O}(\sqrt{n}m)$, an input of minimum size can be computed efficiently.

Similar results about strong controllability have been proved in [19]. These are presented in Section 4.2.2.

4.2.2 On strong controllability

In [19], the strong controllability of a system has been characterized from the constrained matchings in the bipartite graph associated with the underlying loop directed graph. The result is the following.

Theorem 4.7. *[19] Let G be a loop directed graph on n nodes with pattern $\boldsymbol{A}$ and S be an input set with cardinality $m \leq n$. System $(\boldsymbol{A}, \boldsymbol{B}(S))$ is strongly S-controllable if and only if $\boldsymbol{A}(S|.)$ has a constrained $(n-m)$-matching and $\boldsymbol{A}_{\times}(S|.)$ has a constrained V_{loop}-less $(n-m)$-matching.*

Given an input set S, in order to check whether or not a system is strongly S-controllable, an $\mathcal{O}(n^2)$ algorithm was presented in [19]. If the system is not strongly S-controllable, the algorithm computes a node set $\tilde{S}$ containing S such that the system is strongly $\tilde{S}$-controllable. However, computing a minimum-size input set for strong controllability is a challenging problem. In that scope, the following theorem has been deduced from Theorem 4.7 [19].

A *self-damped* system is a system where the underlying graph has a loop on each node (the state of each node influences itself).

A maximum constrained self-less t-matching is a constrained self-less t-matching such that there is no constrained self-less s-matching with $s > t$.

Theorem 4.8. *[19] Consider a loop directed graph G on n nodes with pattern $\boldsymbol{A}$ underlying a self-damped system. A node subset $S \subseteq V$ is a (minimum-size) input set for strong controllability of the system if and only if there is a (maximum) constrained self-less matching in $\boldsymbol{A}_{\times}$ such that in the bipartite graph (V^+, V^-, E) associated with $\boldsymbol{A}_{\times}$, the unmatched nodes of V^- correspond to the nodes of S.*

Proof. Since patterns $\mathbf{A}$ and $\mathbf{A}_{\times}$ are equal, the result is deduced from Theorem 4.7. □

This theorem provides a way to obtain a minimum-size input set for strong controllability in the case of a self-damped system. However, computing a maximum constrained self-less matching in a bipartite graph is a challenging problem.

In the next section, we present our results linking strong controllability to zero forcing. On the one hand, we show that testing whether or not a system is strongly S-controllable is equivalent to testing if S is a zero forcing set in a loop directed graph. On the other hand, we show that in the case of a self-damped

system the zero forcing sets in the simple interconnection graph provide input sets for the strong controllability of the system. In particular, this result together with the existing algorithms on zero forcing provide a way to select in polynomial time a minimum-size input set for a self-damped system with a tree-structure.

4.3 Strong controllability and zero forcing sets

While Theorems 4.7 and 4.8 provide criteria for strong controllability in terms of constrained matchings, we provide in this section equivalent criteria in terms of zero forcing sets. On the one hand, these new statements show that given an input set S, testing whether or not the system is strongly S-controllable is equivalent to checking if S is a zero forcing set in a loop directed graph. On the other hand, they show that in the case of a self-damped system, there is a one-to-one correspondence between the input sets providing strong controllability and the zero forcing sets in a simple directed graph. This result together with the existing algorithms on the zero forcing sets solve the problem of efficiently finding a minimum-size input set for the strong controllability of a self-damped system with a tree-structure.

Statements of Theorems 4.7 and 4.8 in terms of zero forcing sets are based on a one-to-one correspondence between the zero forcing sets in a loop directed graph G and the constrained matchings in the bipartite graph associated with G.

Recall the definition of the color change rule in a loop directed graph G: suppose each node of G is either black or white. If exactly one out-neighbor j of node i is white (possibly $j = i$), then change the color of node j to black.

Whereas in a *loop* directed graph, a node does not need to be black to change the color of one of its out-neighbors, in a *simple* directed graph a node needs instead to be black. Here is a reminder of the color change rule in a simple directed graph.

In a simple directed graph G, the color change rule is defined as follows: suppose each node of G to be black or white. If node i is *black* and node j is the only white (out-)neighbor of i, then change the color of j to black.

Definition 4.9. *[47] Let G be a simple/loop directed graph.*

- *Suppose that each node of G is either black or white. When the color change rule is applied to node i to change the color of node j, we say that i forces j and*

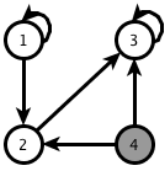


Figure 4.2: A loop directed graph with initially all the nodes white except node 4. A chronological list of forces is: $2 \to 3, 4 \to 2$ and $1 \to 1$.

write $i \to j$.

- *Given a zero forcing set of G, we can list the forces in order in which they were performed to color the nodes of G in black. This list is called a chronological list of forces.*

As an example, consider the loop directed graph in Figure 4.2 with all the nodes initially white, except node 4. A chronological list of forces is then: $2 \to 3, 4 \to 2$ and $1 \to 1$.

Notice that given a zero forcing set, a chronological list of forces is not necessarily unique. Indeed, in the example of Figure 4.2, another chronological list of forces is: $3 \to 3, 4 \to 2$ and $1 \to 1$. However, uniqueness is not required here.

Theorem 4.10. *[40] Let $B = (V^+, V^-, E)$ be a bipartite graph and $\mathscr{M}$ be a matching in B. The following assertions are equivalent:*

- *$\mathscr{M}$ is a constrained matching*
- *We can order the nodes of V^+, $i_1, ..., i_n$ and the nodes of V^-, $j_1, ..., j_n$ such that for each $1 \leq k \leq |\mathscr{M}|$, $\{i_k, j_k\} \in \mathscr{M}$ and for each $1 \leq l < k \leq |\mathscr{M}|$, $\{i_l, j_k\} \notin E$.*

As a consequence of this theorem, we suppose w.l.o.g. that given a constrained matching $\mathscr{M}$ in a bipartite graph (V^+, V^-, E), the nodes of V^+ and V^- are ordered so that for each $1 \leq k \leq |\mathscr{M}|$, $\{i_k, j_k\} \in \mathscr{M}$ and

$$\text{for each } 1 \leq l < k \leq |\mathscr{M}|, \quad \{i_l, j_k\} \notin E \tag{4.2}$$

In Definition 2.31 of the bipartite graph $B_G = (V^+, V^-, E)$ associated with a loop directed graph G, the nodes of V^+ are denoted by $1^+, ..., n^+$ whereas the nodes of V^- are denoted by $1^-, ..., n^-$. However, in this section, Theorem 4.10 suggests we re-order the nodes in V^+ and V^-. Therefore, in what follows, we prefer to denote the nodes of V^+ by $i_1, ..., i_n$ and the nodes of V^- by $j_1, ..., j_n$.

The definition of the bipartite graph $B_G = (V^+, V^-, E)$ associated with a loop directed graph G becomes then: the node sets $V^+ = \{i_1, ..., i_n\}$ and $V^- = \{j_1, ..., j_n\}$ are two copies of the node set of G. Moreover, $\{i_k, j_l\}$ is an edge of B_G if and only if there is an edge from node i_k to node j_l in G.

Lemma 4.11. *Let G be a loop directed graph with node set V, $B_G = (V^+, V^-, E)$ be its associated bipartite graph and $\mathscr{M} := \{\{i_1, j_1\}, ..., \{i_t, j_t\}\}$ be a constrained matching in B_G. Then, $V \backslash \{j_1, ..., j_t\}$ is a zero forcing set in G with chronological list of forces $i_1 \to j_1, ..., i_t \to j_t$.*

Proof. Start with the initial coloring of G where nodes $j_1, ..., j_t$ are the only white nodes. From the definition of the bipartite graph B_G, we know that in G node j_1 is a out-neighbor of node i_1, since $\{i_1, j_1\} \in E$. Moreover, from property (4.2), we know that in G node j_1 is the only white out-neighbor of node i_1. Therefore, i_1 forces j_1. By iterating this argument on all the edges $\{i_2, j_2\}, ..., \{i_t, j_t\}$ of $\mathscr{M}$, we prove that $V \backslash \{j_1, ..., j_t\}$ is a zero forcing set of G. □

Theorem 4.12. *Let G be a loop directed graph with node set V and $B_G = (V^+, V^-, E)$ be the bipartite graph associated with G. Then, $V \backslash \{j_1, ..., j_t\}$ is a zero forcing set of G with a chronological list of forces $i_1 \to j_1, i_2 \to j_2, ..., i_t \to j_t$ if and only if $\mathscr{M} := \{\{i_1, j_1\}, \{i_2, j_2\}, ..., \{i_t, j_t\}\}$ is a constrained matching in B_G.*

Proof. The sufficient condition has been proved in Lemma 4.11. Suppose that $V \backslash \{j_1, ..., j_t\}$ is a zero forcing set of G with chronological list of forces

$$i_1 \to j_1, ..., i_t \to j_t.$$

For each $1 \leq l \leq t$, since $i_l \to j_l$, node j_l is a out-neighbor of node i_l. Therefore, $\{i_l, j_l\}$ is an edge of B_G. In addition, since each white node is forced only once and each node forces at most one node, $\mathscr{M} := \{\{i_1, j_1\}, \{i_2, j_2\}, ..., \{i_t, j_t\}\}$ is a matching in B_G. Since i_1 forces j_1, node j_1 is the only white out-neighbor of node i_1. Hence, for each $1 < k \leq t$, $\{i_1, j_k\} \notin E$. By iterating this argument on all the forces $i_2 \to j_2, ..., i_t \to j_t$, we prove that $\mathscr{M}$ meets property (4.2). Therefore, from Theorem 4.10, $\mathscr{M}$ is a constrained matching. □

Notice that in the previous theorem the set $V \backslash \{j_1, ..., j_t\}$ is the set of unmatched nodes in the node subset V^- of B_G resulting from the constrained matching $\mathscr{M}$.

Thanks to the previous result, we can re-state Theorem 4.7 in terms of zero forcing sets.

Remember that V_{loop} is the set of nodes with a loop in the underlying loop directed graph G.

Theorem 4.13 (Restatement of Theorem 4.7)**.** *Let G be a loop directed graph on n nodes with pattern $\boldsymbol{A}$ and S be an input set with cardinality $m \leq n$. System $(\boldsymbol{A}, \boldsymbol{B}(S))$ is strongly S-controllable if and only if*

- *S is a zero forcing set of G and*

- *S is a zero forcing set of $G_\times$ for which there is a chronological list of forces that does not contain a force of the form $i \to i$ with $i \in V_{loop}$.*

Proof. Theorem 4.12 states that $\mathbf{A}(S|.)$ has a constrained $(n-m)$-matching if and only if S is a zero forcing set of G. Indeed, in the bipartite graph (V^+, V^-, E) associated with $\mathbf{A}$, the unmatched nodes of V^- resulting from a constrained $(n-m)$-matching of $\mathbf{A}(S|.)$ are the nodes in S.

From the same argument, $\mathbf{A}_\times(S|.)$ has a constrained $(n-m)$-matching if and only if S is a zero forcing set of $G_\times$. Besides, there is a constrained $(n-m)$-matching which is V_{loop}-less in $\mathbf{A}_\times(S|.)$ if and only if in $G_\times$ zero forcing set S has a chronological list of forces with no force of the form $i \to i$ with $i \in V_{loop}$. Indeed, Theorem 4.12 states that the edges in the constrained matching form a chronological list of forces for zero forcing set S.

Theorem 4.7 concludes the proof. □

From Theorem 4.13, notice that testing if a system is strongly S-controllable is equivalent to checking if S is a zero forcing set in a loop directed graph. As an example, consider the system for which the underlying loop directed graph G is in Figure 4.3 and check that this system is strongly S-controllable for $S = \{1\}$. We immediately check that S is a zero forcing set of G and that S is a zero forcing set of $G_\times$ with chronological list of forces: $1 \to 2, 2 \to 3$. Since $V_{loop} = \{1\}$ and in this list node 1 does not force itself, the system is then strongly S-controllable.

Moreover, notice that checking if a node subset S is a zero forcing set in a loop directed graph G on n nodes can be done in $\mathcal{O}(n^2)$ time. Indeed, here is an example of algorithm: we start by listing the white nodes in the initial coloring of G and we connect each node of G with its white out-neighbors in this list. At each iteration, we check each node of G one by one in order to find a node i that has exactly one white out-neighbor j. If we find such a node i, then we remove node j from the list of the white nodes and we iterate again. The algorithm ends when there is no node with exactly one white out-neighbor any more. The set S is a zero forcing set if and only if the list of the white

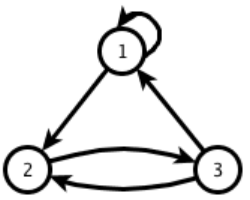


Figure 4.3: A system with this loop directed graph as underlying structure is strongly S-controllable with $S = \{1\}$.

nodes is empty when the algorithm ends. Since there are $\mathcal{O}(n)$ iterations that cost $\mathcal{O}(n)$ time each, the algorithm runs in $\mathcal{O}(n^2)$ time.

As a consequence, Theorem 4.13 provides an intuitive quadratic-time algorithm to check is a system is strongly S-controllable.

The following lemma is a first step to a statement of Theorem 4.8 in terms of zero forcing sets in a *simple* directed graph.

Lemma 4.14. *Consider a loop directed graph G on n nodes underlying a self-damped system. A node subset $S \subseteq V$ is an input set for strong controllability of the system if and only if S is a zero forcing set in $G_\times$ for which there is a chronological list of forces with no force of the form $i \to i$.*

Proof. Since the system is self-damped, $G_\times = G$ and $V_{loop} = V$. Consequently, this result follows from Theorem 4.13. □

From this lemma, we deduce a statement of Theorem 4.8 in terms of zero forcing sets in a *simple* directed graph. The simple directed graph G_s[‡] is obtained from the loop directed graph G by removing the loops on its nodes.

Recall that in a simple directed graph, a node must be black to be able to force one of its out-neighbors.

Theorem 4.15 (Restatement of Theorem 4.8)**.** *Consider a loop directed graph G on n nodes underlying a self-damped system. A node subset $S \subseteq V$ is a (minimum-size) input set for strong controllability of the system if and only if S is a (minimum) zero forcing set in the simple directed graph G_s.*

Proof. We show that S is a zero forcing set in G_s if and only if S is a zero forcing set in $G_\times$ for which there is a chronological list of forces with no force of the

[‡]Whenever s is used as a subscript, it always refers to a simple directed graph. The node subset S is never used as a subscript.

form $i \to i$. Indeed, in $G_\times$ each node has a loop. Moreover, if there is no force of the form $i \to i$, it means that each node must be black to be able to force one of its out-neighbors. Therefore, S is a zero forcing set in G_s if and only if S is a zero forcing set in $G_\times$ for which there is a chronological list of forces with no force of the form $i \to i$.

This observation together with Lemma 4.14 proves the theorem. □

This shows that the minimum zero forcing sets in the simple directed graph G_s provide minimum-size input sets for strong controllability of the underlying self-damped system. However, we deduce from the NP-hardness result of Aazami [1] about the zero forcing number of a simple undirected graph that computing a minimum zero forcing set in a simple directed graph is an NP-hard problem. Nevertheless, there are some efficient algorithms computing minimum zero forcing sets in a simple tree, such as Algorithm 2. Such algorithms together with the previous theorem show that we can compute efficiently a minimum-size input set for strong controllability of a self-damped system with a tree structure.

Theorem 4.16. *A minimum-size input set for strong controllability of a self-damped system with a tree-structure can be computed in polynomial time.*

4.4 Related results

In 2014, a result related to Theorems 4.13 and 4.15 was found by Monshizadeh *et al.* [77], but for systems underlying a *simple* directed graph. In this section, we explain the main result of [77] and compare it with our results from the previous section.

A simple directed graph $G_s = (V, E)$ on n nodes defines the matrix family

$$\mathscr{Q}_{sd}(G_s) = \{A \in \mathbb{R}^{n\times n} : \text{ for } i \neq j, a_{ij} \neq 0 \Leftrightarrow (j,i) \in E\}.$$

The pattern $\mathbf{A}_s$ associated with G_s is an $n \times n$ matrix, where an off-diagonal entry a_{ij} is a star $\star$ if and only if (j,i) is an edge of G_s, every diagonal entry is a question mark and the other entries are zero. As an example, the pattern $\mathbf{A}_s$ of the simple directed graph G_s in Figure 4.4(b) is

$$\mathbf{A}_s = \begin{pmatrix} ? & \star & 0 \\ \star & ? & 0 \\ \star & \star & ? \end{pmatrix}.$$

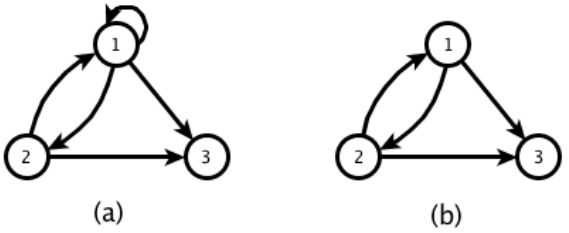


Figure 4.4: A loop directed graph G in (a) and its associated simple directed graph G_s in (b). Given the input set $S = \{1\}$, system $(\mathbf{A}, \mathbf{B}(S))$ underlying G is strongly S-controllable, whereas system $(\mathbf{A}_s, \mathbf{B}(S))$ underlying G_s is not.

A star $\star$ denotes a nonzero entry, whereas a question mark can be a zero or nonzero entry. The following matrices

$$A_1 = \begin{pmatrix} -3 & 1 & 0 \\ 9 & 0 & 0 \\ -5 & -4 & 0 \end{pmatrix} \quad A_2 = \begin{pmatrix} 0 & 1 & 0 \\ 2 & -3 & 0 \\ 1 & -4 & 8 \end{pmatrix}$$

are both in $\mathscr{Q}_{sd}(G_s)$ and are two realizations of pattern $\mathbf{A}_s$. We write $A_1 \in \mathbf{A}_s$ and $A_2 \in \mathbf{A}_s$.

Unlike the matrices of $\mathscr{Q}_{ld}(G)$ defined by a loop directed graph G, the matrices in $\mathscr{Q}_{sd}(G_s)$ have a free diagonal. For example, have a look at the loop directed graph G in Figure 4.4(a) for which the associated simple directed graph is in Figure 4.4(b). The pattern $\mathbf{A}$ associated with G is

$$\mathbf{A} = \begin{pmatrix} \star & \star & 0 \\ \star & 0 & 0 \\ \star & \star & 0 \end{pmatrix}.$$

Therefore, while A_1 and A_2 are both a realization of $\mathbf{A}_s$, A_1 is a realization of $\mathbf{A}$ whereas A_2 is not.

Consequently, given a loop directed graph G and its associated simple directed graph G_s,

$$\mathscr{Q}_{ld}(G) \subseteq \mathscr{Q}_{sd}(G_s).$$

A networked system for which the underlying graph is a *simple* directed

graph $G_s = (V, E)$ on n nodes is a linear system of the form:

$$\dot{x}(t) = Ax(t) + Bu(t),$$

where $A \in \mathscr{Q}_{sd}(G_s)$, $x(t) \in \mathbb{R}^n$ is the state vector, $B \in \mathbf{B}(S)$ for some node set $S \subseteq V$ of size m and $u(t) \in \mathbb{R}^m$ is the outside controller. Such a system is referred to as system (A, B).

Given an input set S, each system (A, B) underlying a simple directed graph G_s with pattern $\mathbf{A}_s$ is a realization of the pair $(\mathbf{A}_s, \mathbf{B}(S))$, meaning that $A \in \mathbf{A}_s$ and $B \in \mathbf{B}(S)$. We write $(A, B) \in (\mathbf{A}_s, \mathbf{B}(S))$. The set of systems underlying G_s is referred to as system $(\mathbf{A}_s, \mathbf{B}(S))$.

Similarly to the systems where the underlying graph is a loop directed graph, we say that a system $(\mathbf{A}_s, \mathbf{B}(S))$ is strongly S-controllable if all the systems $(A, B) \in (\mathbf{A}_s, \mathbf{B}(S))$ are controllable.

Here is the main result of [77] about strong controllability of system $(\mathbf{A}_s, \mathbf{B}(S))$.

Theorem 4.17. *[77] Given a simple directed graph $G_s = (V, E)$ with pattern $\boldsymbol{A}_s$ and an input set $S \subseteq V$, system $(\boldsymbol{A}_s, \boldsymbol{B}(S))$ is strongly S-controllable if and only if S is a zero forcing set of G_s.*

For a subset $\mathscr{Q}'(G_s) \subseteq \mathscr{Q}_{sd}(G_s)$, a system subset of $(\mathbf{A}_s, \mathbf{B}(S))$ is the set of systems (A, B) where $A \in \mathscr{Q}'(G_s)$ and $B \in \mathbf{B}(S)$.

Such a system subset is strongly S-controllable if all the systems (A, B) with $A \in \mathscr{Q}'(G_s)$ and $B \in \mathbf{B}(S)$ are controllable.

In particular, if G_s is the simple directed graph with pattern $\mathbf{A}_s$ associated with a loop directed graph G with pattern $\mathbf{A}$, the system set $(\mathbf{A}, \mathbf{B}(S))$ is a subset of the systems in $(\mathbf{A}_s, \mathbf{B}(S))$ since $\mathscr{Q}_{ld}(G) \subseteq \mathscr{Q}_{sd}(G_s)$.

By definition, if system $(\mathbf{A}_s, \mathbf{B}(S))$ is strongly S-controllable, then each system subset of $(\mathbf{A}_s, \mathbf{B}(S))$ is strongly S-controllable. However, the converse is false in general.

As an example, take the input set $S = \{1\}$ and consider the system $(\mathbf{A}, \mathbf{B}(S))$ where the underlying graph is the loop directed graph G in Figure 4.4(a) and system $(\mathbf{A}_s, \mathbf{B}(S))$ where the underlying graph is the simple directed graph G_s in Figure 4.4(b) associated with G. From Theorem 4.13, we deduce that $(\mathbf{A}, \mathbf{B}(S))$ is strongly S-controllable since S is a zero forcing set of G and a zero forcing set of $G_\times$ where node 1 does not need to force itself. Instead, since S is not a zero forcing set in the simple directed graph G_s, Theorem 4.17 claims that $(\mathbf{A}_s, \mathbf{B}(S))$ is not strongly S-controllable.

Consequently, given a loop directed graph G with pattern $\mathbf{A}$ and its associated simple directed graph G_s with pattern $\mathbf{A}_s$, Theorem 4.13 analyzes the strong controllability of system $(\mathbf{A}, \mathbf{B}(S))$, for some input set S whereas Theorem 4.17 is about the strong controllability of a bigger system set $(\mathbf{A}_s, \mathbf{B}(S))$ underlying G_s.

In addition, from Theorems 4.15 and 4.17, we deduce the following result.

Theorem 4.18. *Let G be a loop directed graph with pattern $\boldsymbol{A}$ underlying a self-damped system and G_s be its associated simple directed graph with pattern $\boldsymbol{A}_s$. System $(\boldsymbol{A}, \boldsymbol{B}(S))$ is strongly S-controllable if and only if system $(\boldsymbol{A}_s, \boldsymbol{B}(S))$ underlying the simple directed graph G_s is strongly S-controllable.*

Other system subsets of $(\mathbf{A}_s, \mathbf{B}(S))$ were studied in [77], notably when the simple directed graph is symmetric. We refer the reader to Section IV.A of [77] for more details.

4.5 Conclusion

Structural controllability studies our ability to control a networked system without any knowledge of the interaction strengths along the edges of the interconnection graph. In particular, strong (structural) controllability checks if a system is controllable whatever the interaction strengths.

The goal of this chapter was to link the notion of zero forcing, constrained matching and strong controllability.

At first, we have proved a one-to-one correspondence between the zero forcing sets in a loop directed graph G and the constrained matchings in the bipartite graph associated with G. Our following contributions on strong controllability were based on this correspondence. On the one hand, we have shown that testing whether or not a system is strongly S-controllable from an input set S is equivalent to checking if S is a zero forcing set in a loop directed graph. Thanks to this result, we have now an intuitive quadratic-time algorithm that checks if a system is strongly S-controllable. On the other hand, we have shown that the (minimum) zero forcing sets in the *simple* interconnection graph provide (minimum-size) input sets for the strong controllability of a self-damped system. In particular, we have deduced that one can find in polynomial time a minimum-size input set for the strong controllability of a self-damped system with a tree structure, using existing algorithms on zero forcing.

All of these results were published in [94]:

M. Trefois and J.-C. Delvenne, *Zero forcing number, constrained matchings and strong structural controllability*, Linear Algebra and Its Applications, 484: 199-218, 2015.

A similar work was done in [77] when the underlying graph is a *simple* directed graph and in [15] where the link between the zero forcing sets of a simple undirected graph and the controllability of a quantum system has been shown.

All these results show the role of the zero forcing sets in the study of the dynamics of networked systems and should motivate additional research on zero forcing.

Chapter 5

Factorizations of the all ones matrix

This chapter is about the problem of factorizing the $n \times n$ matrix

$$J_n = \begin{pmatrix} 1 & 1 & \dots & 1 \\ 1 & 1 & \dots & 1 \\ \vdots & \vdots & & \vdots \\ 1 & 1 & \dots & 1 \end{pmatrix}$$

into binary matrices. Precisely, we restrict ourselves to square $n \times n$ factors A_i that have all their entries in $\{0,1\}$, i.e. that are adjacency matrices of loop directed graphs with n nodes. We are thus looking for the solutions of

$$\prod_{i=1}^{m} A_i = A_1 A_2 ... A_m = J_n$$

and in particular, we are interested in the case when all the factors are identical, i.e. in the binary solutions to the equation

$$A^m = J_n.$$

The binary solutions to $A^m = J_n$ are of interest in different areas. Here are some examples of application:

- the solutions of $A^m = J_n$ can be seen as the adjacency matrices of the loop directed graphs for which there is a unique walk of length m from any node i to any node j. In [73], it has been shown that these graphs can be used to model algebras defined from a set S that is the cartesian product of m copies of a given finite set and from the $m-1$ binary operators $f_1, ..., f_{m-1}$ on S defined by:

$$f_i((a_1, a_2, ..., a_m), (b_1, b_2, ..., b_m)) = (a_{i+1}, a_{i+2}, ..., a_m, b_1, b_2, ..., b_i).$$

The nodes of the loop directed graph modelling such an algebra are the elements of S and the $m-1$ operators $f_1,...,f_{m-1}$ applied on the nodes s_k and s_l are modelled by the unique walk of length m from s_k to s_l. For example, if $m=3$ and if the unique path of length 3 from s_k to s_l is

$$s_k \to x \to y \to s_l,$$

then x and y are uniquely determined by s_k and s_l and we have $f_1(s_k,s_l)=x$ and $f_2(s_k,s_l)=y$.

- recall that the finite time average consensus problem can be described as follows: we start with a number n of agents which all have an initial position, unknown to the agents. At each step, each agent receives a limited number of positions and with the received information it will change its own position. The goal is that all the agents meet at the average of their initial positions after a finite number of steps. The binary solutions to $A^m = J_n$ represent communication topologies where the agents all meet after m steps. Moreover, in these strategies, each agent receives at each step a number p of positions such that $n = p^m$. It has been shown in [29] that these strategies are the quickest among all the communication topologies where each agent receives p positions per step.

The De Bruijn matrices, which are a family of solutions to $A^m = J_n$, will be widely used throughout this chapter. Such a matrix is the adjacency matrix of a De Bruijn graph, originally defined in [26]. A De Bruijn graph has $n = p^m$ nodes which are all the possible sequences of length m of p given symbols. Regarding the edges, if a node i can be obtained by shifting the sequence of symbols of a node j of one position to the right by removing the last symbol of j and then by adding one symbol at the beginning of the sequence, then node j is a out-neighbor of node i. Typically, we have:

$$i=(s_1,s_2,...,s_m) \to j=(s_2,...,s_m,s).$$

The De Bruijn graph where the nodes are all the sequences of length 3 of two symbols a and b is given in Figure 5.1.

Most of the literature [59,68,100,101] about the binary solutions to $A^m = J_n$ assumes the matrix A to be g-circulant, i.e. each row of A is obtained from the previous one by shifting all its entries g positions to the right. Some of the g-circulant solutions to $A^m = J_n$ have been proved in [101] to be isomorphic to a De Bruijn matrix.

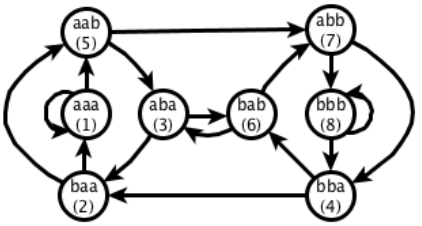


Figure 5.1: The De Bruijn graph where the nodes are all the sequences of length 3 of two symbols a and b.

Although the De Bruijn matrices are the only well known binary roots of J_n, it has been shown [69] that each binary solution to $A^m = J_n$ shares several properties with the De Bruijn matrices. The authors of [69] even call the binary solutions to $A^m = J_n$ *Generalized De Bruijn matrices*. However, the relation between the De Bruijn matrices and the binary solutions to $A^m = J_n$ is still far from being fully understood.

The goal of this chapter is to improve our understanding of the binary solutions to $A^m = J_n$ and in particular to show how the binary roots of J_n with minimum rank are related to the De Bruijn matrices.

Section 5.1 proves some straightforward properties on the binary roots of J_n and presents the De Bruijn matrices.

Section 5.2 is about the commuting factors of the matrix with all ones. We prove that under some conditions on the commuting factors, these are isomorphic to a row permutation of a matrix having the same structure as a De Bruijn matrix. Moreover, we give two non trivial examples of matrices that show that the De Bruijn matrices are not the only matrix roots of J_n, up to isomorphism.

In Section 5.3, we prove that each binary root of J_n with minimum rank is isomorphic to a row permutation of a De Bruijn matrix. From this result, we deduce a characterization of the binary solutions to $A^2 = J_n$ with minimum rank, which partially solves Hoffman's open problem of characterizing the binary solutions to $A^2 = J_n$.

In Section 5.4, we provide a class of roots, not necessarily g-circulant, which are isomorphic to a De Bruijn matrix.

Finally, Section 5.5 concludes this chapter.

Throughout the sections of this chapter, the rows and columns of a matrix of size n will be indexed, for convenience, from 0 to $n-1$ and the vector $\mathbf{1}_n$

will denote the $n \times 1$ vector of all 1's.

Moreover, an $n \times n$ binary matrix A is seen as the adjacency matrix of a loop directed graph G on n nodes. When G is p-regular (recall that this means that each node of G has out- and in-degree equal to p), we say by abuse of language that A is p-regular, which can be expressed as:

$$A\mathbf{1}_n = A^T\mathbf{1}_n = p\mathbf{1}_n.$$

Our contributions for this topic were published in [96]:

M. Trefois, P. Van Dooren and J.-C. Delvenne, *Binary factorizations of the matrix of all ones*, Linear Algebra and Its Applications, 468: 63-79, 2015.

5.1 Matrix roots of J_n and De Bruijn matrices

In this section, we prove some properties of the binary roots of the square matrix J_n of all ones. Moreover, we present the De Bruijn matrices and we show they are binary roots of J_n.

Lemma 5.1. *Let $A^m = kJ_n$, where $k \neq 0$ and $A \in \{0,1\}^{n\times n}$, then A is p-regular, the trace of A is p, p and k are positive integers and $p^m = kn$.*

Proof. Clearly, the spectrum of kJ_n is given by

$$\Lambda(kJ_n) = \{kn, 0, ..., 0\}.$$

Moreover, there is an invertible matrix $U \in \mathbb{C}^{n\times n}$ and a triangular matrix $T \in \mathbb{C}^{n\times n}$ such that

$$UAU^{-1} = T$$

and the diagonal $\lambda_1, ..., \lambda_n$ of T is the spectrum of A. As a consequence,

$$U(kJ_n)U^{-1} = UA^mU^{-1} = T^m$$

and the diagonal $\lambda_1^m, ..., \lambda_n^m$ of T^m is the spectrum of kJ_n.

Therefore, the spectrum of an m^{th} root A of kJ_n is:

$$\Lambda(A) = \{p, 0, ..., 0\},$$

with $p^m = kn$. Since A is a binary matrix, its trace, which is equal to p, must be a nonnegative integer. Moreover, the entries of A^m must be equal to k, which

is therefore also a nonnegative integer. Since we ruled out $k = 0$, both p and k must be positive integers. Since p is then the only strictly positive eigenvalue of A, its left and right Perron vectors must be proportional to $\mathbf{1}_n$, i.e.

$$A\mathbf{1}_n = p\mathbf{1}_n \ \ and \ \ A^T\mathbf{1}_n = p\mathbf{1}_n,$$

which implies that A is p-regular. □

The De Bruijn matrices are a well known example of p-regular binary matrices with a trace equal to p. Below, we define the De Bruijn matrices and we show they are binary roots of J_n.

In the introduction of this chapter, we have presented the De Bruijn matrices as being the adjacency matrices of the De Bruijn graphs. We give now a matricial definition of a De Bruijn matrix.

Two matrices A and B are said to be *isomorphic* if there is a permutation matrix P such that $PAP^T = B$. It should be mentioned that a permutation matrix P is orthogonal and therefore $P^{-1} = P^T$.

Definition 5.2. *The De Bruijn matrix of order p and size $n = p^m$ (for some integer m) is, up to isomorphism, the $n \times n$ matrix defined as:*

$$D(p,n) := \mathbf{1}_p \otimes I_{n/p} \otimes \mathbf{1}_p^T,$$

where $I_{n/p}$ is the identity matrix of size n/p, $\mathbf{1}_p$ is the $p \times 1$ vector with all ones and $\otimes$ denotes the Kronecker product defined as follows: given an $n \times m$ matrix $A = (a_{ij})_{n\times m}$ and a $p \times q$ matrix B, the Kronecker product of A and B, denoted by $A \otimes B$, is the $np \times mq$ matrix given by:

$$A \otimes B = \begin{pmatrix} a_{11}B & \dots & a_{1m}B \\ \vdots & & \vdots \\ a_{n1}B & \dots & a_{nm}B \end{pmatrix}.$$

As an example, the De Bruijn matrix $D(2,8)$ of order 2 and size 8 is:

$$D(2,8) = \begin{pmatrix} 1 & 1 & & & & & & \\ & & 1 & 1 & & & & \\ & & & & 1 & 1 & & \\ & & & & & & 1 & 1 \\ 1 & 1 & & & & & & \\ & & 1 & 1 & & & & \\ & & & & 1 & 1 & & \\ & & & & & & 1 & 1 \end{pmatrix}.$$

We notice that this matrix corresponds to the adjacency matrix of the De Bruijn graph in Figure 5.1.

Below, we prove that the De Bruijn matrix $D(p,n)$ (with $n = p^m$) is the adjacency matrix of the De Bruijn graph on p^m nodes.

In a graph, *reversing an edge* (i,j) means replacing it by the edge (j,i).

Given a graph G, the *transpose* of G, denoted by G^T, is the graph obtained from G by reversing all its edges.

Lemma 5.3. *If G is a De Bruijn graph, then G and G^T are isomorphic.*

Proof. Suppose $G = (V,E)$ is the De Bruijn graph on p^m nodes. We define the following bijection between the nodes of G and G^T:

$$f : V \to V : (s_1, s_2, ..., s_{m-1}, s_m) \mapsto (s_m, s_{m-1}, ..., s_2, s_1).$$

We now have to prove that given two nodes i and j, there is an edge in G from i to j if and only if in G^T there is an edge from $f(i)$ to $f(j)$.

In G, there is an edge from i to j if and only if there is a sequence $x = (s_1, ..., s_{m-1})$ of $m-1$ symbols such that $i = (s,x)$ and $j = (x,s')$. From the definition of the above bijection f, we deduce that in G there is an edge from i to j if and only if G has an edge from $f(j) = (s', x^{-1})$ to $f(i) = (x^{-1}, s)$, where $x^{-1} = (s_{m-1}, ..., s_1)$. Since G^T is obtained by reversing the edges of G, the result is proved. □

Recall that throughout this chapter, the rows and columns of a matrix of size n are indexed, for convenience, from 0 to $n-1$.

Proposition 5.4. *The De Bruijn matrix $D(p,n)$ (with $n = p^m$) is the adjacency matrix of the De Bruijn graph on p^m nodes.*

Proof. From the definition of $D(p,n)$ (cf. Definition 5.2), it follows that the (i,j)-entry of $D(p,n)$ is 1 if and only if

$$\left\lfloor \frac{j}{p} \right\rfloor = i \mod \frac{n}{p}.$$

Moreover, if we consider the nodes of the De Bruijn graph to be all the numbers from 0 to $p^m - 1$ written in basis p, we have that:

- if i in basis p is $i = s_1 s_2 ... s_m$, then $i \mod \frac{n}{p}$ in basis p is $s_2 ... s_m$,

- if j in basis p is $j = s_1 ... s_{m-1} s_m$, then $\left\lfloor \frac{j}{p} \right\rfloor$ in basis p is $s_1 ... s_{m-1}$.

Therefore, we deduce that the (i,j)-entry of $D(p,n)$ is 1 if and only if there is an edge from i to j in the De Bruijn graph.

As a consequence, we have proved that $D(p,n)$ is the adjacency matrix of the *transpose* of the De Bruijn graph. However, since Lemma 5.3 states the De Bruijn graph and its transpose are isomorphic, it follows that $D(p,n)$ is the adjacency matrix of the De Bruijn graph. □

It should be noted that the rank of the De Bruijn matrix $D(p,n)$ is n/p.

The following lemma is a well-known result about the De Bruijn matrices.

Lemma 5.5. *The i^{th} power of the De Bruijn matrix $D(p,n)$ (with $n = p^m$) is equal to*

$$D(p,n)^i = D(p^i, n) = \mathbf{1}_{p^i} \otimes I_{p^{m-i}} \otimes \mathbf{1}_{p^i}^T$$

for $i \leq m$ and equal to

$$D(p,n)^i = p^{i-m} J_n$$

for $i \geq m$.

A direct consequence of this lemma is the following.

Corollary 5.6. *The De Bruijn matrices $D(p,n)$ are such that*

$$D(p,n)^m = J_n, \quad \forall\ n = p^m,$$
$$D(p,n)^m = kJ_n, \quad \forall\ kn = p^m.$$

The De Bruijn matrices $D(p,n)$ for which $n = p^m$ are thus m^{th} roots of J_n. In the next section, we will show that under a certain condition on the rank of

the m^{th} roots of J_n, these are isomorphic to a row permutation of a De Bruijn matrix.

By abuse of notation, given two integers n and p such that p divides n (n not necessarily a power of p), the matrix

$$\mathbf{1}_p \otimes I_{n/p} \otimes \mathbf{1}_p^T$$

will be denoted by $D(p,n)$.

5.2 Factorizations into commuting factors

In this section, we consider the factorizations of J_n into two commuting factors A and B that are required to be regular, namely

$$AB = BA = J_n, \tag{5.1}$$

where $A, B \in \{0,1\}^{n\times n}$, A is p-regular and B is l-regular.

The goal of what follows is to prove Theorem 6.3 claiming that, under a certain condition on their rank, such factors are isomorphic to a row permutation of the matrix $D(p,n)$ or $D(l,n)$. In that scope, the following terminology is used.

Let P_1 and P_2 be two permutation matrices. When we say that *the permutations of P_1 are absorbed in P_2*, this means that we denote by P_2 the permutation matrix P_1P_2.

The notation $A[a:b, c:d]$ refers to the submatrix of A with the rows of A indexed from a to b and the columns of A indexed from c to d.

Recall that two matrices A and B are said to be *isomorphic* if there is a permutation matrix P such that $PAP^T = B$. We write $A \cong B$.

In the following, $e_{i,n}$ ($0 \le i \le n-1$) denotes the $n \times 1$ unit vector with 1 in its i^{th} position.

Theorem 5.7. *Let A and B be two regular matrices satisfying (5.1), then $pl = n$. Moreover,* $\mathrm{rank}(A) = n/p$ *(resp.* $\mathrm{rank}(B) = n/l$*) if and only if there is a permutation matrix P such that*

$$A \cong PD(p,n) \quad (resp.\ B \cong PD(l,n)).$$

Since the proof of the above theorem is somewhat technical, we first give the guidelines: by using the three hypothesis that A is p-regular, B is l-regular and that $AB = BA = J_n$, we perform row and column permutations on A and B in order to obtain two matrices isomorphic to A and B respectively where one of the two is made up of p rows of p identity matrices I_l. Since such a matrix is isomorphic to $D(p,n)$, one of A and B is isomorphic to $D(p,n)$ and by symmetry the other factor is isomorphic to $D(l,n)$.

Proof. (of Theorem 5.7) First of all, notice that we can suppose without loss of generality that the entries a_{00} and b_{00} of A and B, respectively, both equal 1. Indeed, since $BA = J_n$, there exist i, j such that $b_{ij}a_{ji} = 1$. Therefore, there are permutation matrices P_i, P_j such that the matrices $\tilde{B} = P_i B P_j^T$ and $\tilde{A} = P_j A P_i^T$ have their $(0,0)$-entry equal to 1. Moreover, $\tilde{A}$ and $\tilde{B}$ are binary matrices such that

$$\tilde{A}\tilde{B} = \tilde{B}\tilde{A} = J_n, \quad \tilde{A} \text{ is } p\text{-regular and } \tilde{B} \text{ is } l\text{-regular.}$$

In addition, if $\tilde{A}$ (resp. $\tilde{B}$) is isomorphic to a matrix of the form $PD(p,n)$ (resp. $PD(l,n)$), then this holds too for A (resp. B). Therefore, we can consider that $a_{00} = b_{00} = 1$.

Since each column of B has l nonzero entries, there exists a row permutation P_2 such that the first column of the matrix P_2B has its l first entries equal to 1, namely

$$P_2 B e_{0,n} = \begin{pmatrix} \mathbf{1}_l \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Moreover, since $(AP_2^T)(P_2B) = J_n$, the block A_1 of the l first columns of AP_2^T satisfies

$$A_1 \mathbf{1}_l = \mathbf{1}_n$$

and since AP_2^T is p-regular,

$$\mathbf{1}_n^T A_1 = p\mathbf{1}_l^T.$$

For such a matrix A_1, there is a row permutation P_1 such that

$$P_1 A_1 = \mathbf{1}_p \otimes I_l.$$

This shows that $pl = n$.

Since A and B have their first entry, a_{00} and b_{00} respectively, equal to 1, we

can assume that P_1 and P_2 have their first row equal to $e_{0,n}^T$.

As a consequence, we have permutation matrices P_1 and P_2 such that

$$(P_1AP_2^T)(e_{0,p} \otimes I_l) = P_1A_1 = \mathbf{1}_p \otimes I_l, \quad (P_2BP_1^T)e_{0,n} = e_{0,p} \otimes \mathbf{1}_l$$

and

$$(P_1AP_2^T)(P_2BP_1^T) = (P_2BP_1^T)(P_1AP_2^T) = J_n.$$

From the fact that

$$P_1AP_2^T = \left(\begin{array}{c|c|c|c} I_l & & \cdots & \\ \hline I_l & & & \\ \hline \vdots & & & \\ \hline I_l & & & \end{array}\right),$$

that

$$P_2BP_1^T = \left(\begin{array}{c|c|c|c} 1 & & & \\ \vdots & & \cdots & \\ 1 & & & \\ \hline 0 & & & \\ \vdots & & & \\ 0 & & & \\ \hline & & & \\ \vdots & & & \\ & & & \\ \hline 0 & & & \\ \vdots & & & \\ 0 & & & \end{array}\right),$$

and that $(P_2BP_1^T)(P_1AP_2^T) = J_n$, it follows that in $P_2BP_1^T$, each (i,j)-entry with

$0 \leq i \leq l-1$ and $j > 0$ such that $j = 0 \mod l$ is zero, namely

$$P_2 B P_1^T = \left(\begin{array}{c|c|c|c} 1 & 0 & & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 1 & 0 & & 0 \\ \hline 0 & & & \\ \vdots & & & \\ 0 & & & \\ \hline & & & \\ \vdots & & & \\ \hline 0 & & & \\ \vdots & & & \\ 0 & & & \end{array}\right).$$

As a consequence, since B is l-regular, there is a permutation matrix P which permutes the $n-l$ last rows of $P_2 B P_1^T$ in such a way that

$$PP_2 B P_1^T = \left(\begin{array}{c|c|c|c|c} 1 & 0 & 0 & & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & 0 & 0 & & 0 \\ \hline 0 & 1 & & & \\ \vdots & \vdots & & & \\ 0 & 1 & & & \\ \hline \vdots & 0 & & & \\ & \vdots & & & \\ & 0 & & & \\ \hline \vdots & \vdots & & & \\ \hline 0 & 0 & & & \\ \vdots & \vdots & & & \\ 0 & 0 & & & \end{array}\right).$$

and

$$P_1 A P_2^T P^T = \left(\begin{array}{c|c|c} I_l & \cdots & \\ \hline I_l & & \\ \hline \vdots & & \\ \hline I_l & & \end{array}\right).$$

Again, since

$$(PP_2 B P_1^T)(P_1 A P_2^T P^T) = J_n,$$

each (i, j)-entry of $PP_2 B P_1^T$ with $l \leq i \leq 2l-1$ and $j \neq l$ such that $j = 0 \mod l$

is zero, i.e.

$$PP_2BP_1^T = \left(\begin{array}{c|c|c|c|c} 1 & 0 & 0 & & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & 0 & 0 & & 0 \\ \hline 0 & 1 & 0 & & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 1 & 0 & & 0 \\ \hline \vdots & & & & \\ \hline \vdots & & & & \\ \hline 0 & & & & \\ \vdots & & & & \\ 0 & & & & \end{array}\right).$$

Repeating this process on the $n-2l$ last rows of $PP_2BP_1^T$ and absorbing all the row permutations in P_2, we get permutation matrices P_1 and P_2 such that

$$P_1AP_2^T = \left(\begin{array}{c|c|c|c} I_l & & \cdots & \\ \hline I_l & & & \\ \hline \vdots & & & \\ \hline I_l & & & \end{array}\right),$$

and for each $i = 0 \mod l$, $(P_2BP_1^T)e_{i,n} = e_{i/l,p} \otimes \mathbf{1}_l$, i.e.

$$P_2BP_1^T = \left(\begin{array}{c|c|c|c|c} 1 & 0 & 0 & & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & 0 & 0 & & 0 \\ \hline 0 & 1 & 0 & & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 1 & 0 & & 0 \\ \hline 0 & 0 & 1 & & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 1 & & 0 \\ \hline \vdots & \vdots & & \ddots & \vdots \\ \hline 0 & 0 & & & 1 \\ \vdots & \vdots & & & \vdots \\ 0 & 0 & & & 1 \end{array}\right).$$

In addition, since $(P_2BP_1^T)(P_1AP_2^T) = J_n$, there is a permutation matrix P such that $P_2BP_1^TP^T$ and $PP_1AP_2^T$ have the same structure as $P_2BP_1^T$ and $P_1AP_2^T$

above, but besides

$$e_{0,n}^T(P_2BP_1^TP^T) = e_{0,p}^T \otimes \mathbf{1}_l^T,$$

i.e.

$$P_2BP_1^TP^T = \left(\begin{array}{c|c|c|c|c} 1 \ \cdots \ 1 & 0 & 0 & & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & 0 & 0 & & 0 \\ \hline 0 & 1 & 0 & & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 1 & 0 & & 0 \\ \hline 0 & 0 & 1 & & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 1 & & 0 \\ \hline \vdots & \vdots & & \ddots & \vdots \\ \hline 0 & 0 & & & 1 \\ \vdots & \vdots & & & \vdots \\ 0 & 0 & & & 1 \end{array}\right).$$

We absorb P in P_1.

Since $(P_1AP_2^T)(P_2BP_1^T) = J_n$, every block $P_1AP_2^T[0 : l-1, i : i+l-1]$ ($i = 0 \mod l$) has exactly one nonzero entry in each row and similarly, since $(P_2BP_1^T)(P_1AP_2^T) = J_n$, every block $P_1AP_2^T[0 : l-1, i : i+l-1]$ ($i = 0 \mod l$) has exactly one nonzero entry in each column. Consequently, there is a permutation matrix P such that

$$P_1AP_2^TP^T = \left(\begin{array}{c|c|c|c} I_l & I_l & \cdots & I_l \\ \hline I_l & & & \\ \hline \vdots & & & \\ \hline I_l & & & \end{array}\right).$$

We absorb P in P_2.

It follows from $\operatorname{rank}(A) = l = n/p$ that all blocks in $P_1AP_2^T$ must be I_l. Therefore, we can update P_1 and P_2 so that

$$P_1AP_2^T = D(p,n).$$

Consequently,

$$A = P_2^T(P_2P_1^TD(p,n))P_2$$

and there is a permutation matrix P such that

$$A \cong PD(p,n).$$

Since A and B are commuting factors, we have the same result for B. This concludes the proof. □

From the proof of Theorem 5.7, we notice that two such commuting factors A and B are such that $\text{rank}(A) \geq n/p$ and $\text{rank}(B) \geq n/l$.

In a factorization $AB = BA = J_n$ with $A \in \{0,1\}^{n\times n}$ p-regular and $B \in \{0,1\}^{n\times n}$ l-regular, if the rank of A (resp. B) is n/p (resp. n/l), we say that *A (resp. B) has minimum rank*.

Remark 5.8. *It is possible that commuting factors do not have a minimum rank. The following example, constructed by hand, was not easily found and is therefore one of our contributions. Consider the matrix*

$$A = \begin{pmatrix} 1 & 1 & 1 & & & & & & \\ & & & 1 & 1 & 1 & & & \\ & & & & & & 1 & 1 & 1 \\ & & 1 & 1 & 1 & & & & \\ 1 & 1 & & & & 1 & & & \\ & & & & & & 1 & 1 & 1 \\ 1 & 1 & & & & 1 & & & \\ & & 1 & 1 & 1 & & & & \\ & & & & & & 1 & 1 & 1 \end{pmatrix}.$$

It is a solution to $A^2 = J_9$. However, $\text{rank}(A) = 4 > 9/3$.

Given matrices $Q_1, ..., Q_s \in \mathbb{R}^{n\times n}$, the block diagonal matrix where the diagonal blocks are $Q_1, ..., Q_s$ is denoted by $\text{diag}(Q_1, ..., Q_s)$.

Remark 5.9. *In the framework of Theorem 5.7, we might wonder whether the factors with minimum rank are in particular isomorphic to $D(p,n)$ or $D(l,n)$. The answer is no. It is not straightforward to find a counter-example. The following one, constructed by hand, is thus another of our contributions. The matrix*

$$A = diag(I_9, Q_2, Q_3)D(3,27),$$

with

$$Q_2 = \begin{pmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ & & & & 1 & & & \\ & & & & & & 1 & \\ & & & & & 1 & & \\ & & & & 1 & & & \\ & & & & & & & 1 \end{pmatrix}$$

and

$$Q_3 = \begin{pmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & & & 1 & & \\ & & & & & & 1 & \\ & & & & & & & 1 \\ & & & 1 & & & & \\ & & & & 1 & & & \\ & & & & & & 1 & \end{pmatrix},$$

is a solution to $A^3 = J_{27}$ with a rank equal to $27/3$. However, since $rank(A^2) = 4 \neq 3 = rank(D(3,27)^2)$, A is not isomorphic to $D(3,27)$.

Theorem 5.7 can be generalized to the case of any number of commuting factors.

Corollary 5.10. *Let $\{A_i\}_{i\in I}$ be a finite set of $n \times n$ binary matrices, each of them p_i-regular such that all their products commute, i.e. for each permutations σ_1, σ_2, $\prod_i A_{\sigma_1(i)} = \prod_i A_{\sigma_2(i)}$, and satisfy*

$$\prod_{i\in I} A_i = J_n.$$

Then, $\prod_{i\in I} p_i = n$. Moreover, for each $i \in I$, $\operatorname{rank}(A_i) = n/p_i$ if and only if there is a permutation matrix P such that

$$A_i \cong PD(p_i, n).$$

Proof. First of all, we prove that $\prod_{i\geq 2} A_i$ is a binary matrix. Indeed, on the one hand, it is clear that all the entries of $\prod_{i\geq 2} A_i$ are nonnegative integers. On the

other hand, if the (i,j)-entry of $\prod_{i\geq 2} A_i$ is greater than one, then since A_1 is p_1-regular, column i of A_1 has at least one nonzero entry, say the (k,i)-entry of A is 1, and therefore, the (k,j)-entry of $A_1(\prod_{i\geq 2} A_i)$ is greater than one, which is a contradiction since $\prod_{i\in I} A_i = J_n$.

Moreover, $\prod_{i\geq 2} A_i$ is$\left(\prod_{i\geq 2} p_i\right)$-regular. Indeed,

$$\left(\prod_{i\geq 2} A_i\right)\mathbf{1}_n = p_2\left(\prod_{i\geq 3} A_i\right)\mathbf{1}_n = \ldots = \left(\prod_{i\geq 2} p_i\right)\mathbf{1}_n$$

and we identically show that

$$\mathbf{1}_n^T\left(\prod_{i\geq 2} A_i\right) = \left(\prod_{i\geq 2} p_i\right)\mathbf{1}_n^T.$$

Theorem 5.7 shows the result for A_1. The same argument repeated on each factor completes the proof. □

Theorem 5.7 can also be applied to the particular case of the binary roots of J_n. We thus have the following result.

Corollary 5.11. *Let A be a binary matrix satisfying*

$$A^m = J_n.$$

Then, A is p-regular and $p^m = n$. Moreover, rank$(A) = n/p$ *if and only if there is a permutation matrix P such that*

$$A \cong PD(p,n).$$

Proof. From Lemma 5.1, we know that A is p-regular and that $p^m = n$. With the same argument as in the proof of Corollary 5.10, we can show that A^{m-1} is a binary matrix. Further, since $A\mathbf{1}_n = p\mathbf{1}_n$ and $\mathbf{1}_n^T A = p\mathbf{1}_n^T$, we deduce that A^{m-1} is p^{m-1}-regular. Theorem 5.7 completes the proof. □

From the proof of Theorem 5.7, we have deduced that the rank of an m^{th} root A of J_n is greater than or equal to $\geq n/p$. Remark 5.8 shows that the rank of A may be greater than n/p and Remark 5.9 shows that A may be non isomorphic to the De Bruijn matrix even though A has a rank equal to n/p.

An m^{th} root of J_n ($n = p^m$) with a rank equal to n/p is said to have minimum rank.

Remark 5.12. *Notice that the previous corollary does not provide a full characterization of the roots with minimum rank since not each row permutation of the De Bruijn matrix is a root of J_n. Indeed, the matrix*

$$A = \begin{pmatrix} 1 & 1 & & & & & & \\ & & 1 & 1 & & & & \\ & & & & 1 & 1 & & \\ & & & & & & 1 & 1 \\ 1 & 1 & & & & & & \\ & & & & & & 1 & 1 \\ & & & & 1 & 1 & & \\ & & 1 & 1 & & & & \end{pmatrix}$$

is not a solution to $A^3 = J_8$.

In the following section, we complete the result of Corollary 5.11 by showing that P can always be chosen as being a block diagonal matrix.

5.3 Roots of J_n with minimum rank

From the proof of Theorem 5.7, we have deduced that a binary solution A to the equation $A^m = J_n$ (recall that A is then p-regular) has a rank of at least n/p. Moreover, we have shown in Corollary 5.11 that a binary root with minimum rank, namely with a rank equal to n/p, is isomorphic to a matrix of the form $PD(p,n)$, where P is a permutation matrix. In this section, we show that P can always be chosen as being a block diagonal matrix.

Lemma 5.13. *Let $A \in \{0,1\}^{n\times n}$ be a p-regular matrix such that $A^m = J_n$. If $rank(A) = n/p$, then A is isomorphic to a matrix of the form*

$$\begin{pmatrix} I_{n/p} \\ Q_2 \\ \vdots \\ Q_p \end{pmatrix} \otimes \mathbf{1}_p^T,$$

where each $Q_i \in \{0,1\}^{(n/p)\times(n/p)}$ is a permutation matrix.

Proof. From Lemma 5.1, we know that the trace of A is p. Consequenlty, we can assume without loss of generality that $a_{00} \neq 0$. Indeed, if it is not the case, since A has a nonzero diagonal entry, say the (i,i)-entry of A equals 1, then

there is a permutation matrix P_i such that A is isomorphic to

$$P_i A P_i^T$$

where the $(0,0)$-entry is nonzero. Therefore, in the following, we suppose without loss of generality that $a_{00} \neq 0$.

Since A is p-regular and $a_{00} \neq 0$, there is a permutation matrix P such that

$$PAP^T = \begin{pmatrix} \overbrace{1 \ \dots \ 1}^{p} & 0 \ \dots \ 0 \\ \star \ \dots \ \star & \star \ \dots \ \star \\ \vdots \quad\quad \vdots & \vdots \quad\quad \vdots \\ \star \ \dots \ \star & \star \ \dots \ \star \end{pmatrix}.$$

Moreover, since A^2 is a binary matrix (this can be proved with the same argument as in the proof of Corollary 5.10), the matrix PA^2P^T is also binary. Therefore, since $PA^2P^T = (PAP^T)(PAP^T)$, the matrix PAP^T must be of the form:

$$PAP^T = \begin{pmatrix} p\left\{\begin{pmatrix} \overbrace{1 \ \dots \ 1}^{p} & 0 \ \dots \ 0 \\ 0 \ \dots \ 0 & \star \ \dots \ \star \\ \vdots \quad\quad \vdots & \vdots \quad\quad \vdots \\ 0 \ \dots \ 0 & \star \ \dots \ \star \end{pmatrix}\right. \\ \begin{matrix} \star \ \dots \ \star & \star \ \dots \ \star \\ \vdots \quad\quad \vdots & \vdots \quad\quad \vdots \\ \star \ \dots \ \star & \star \ \dots \ \star \end{matrix} \end{pmatrix}.$$

Again, since PAP^T is p-regular, we can update P so that PAP^T is of the form:

$$PAP^T = \begin{pmatrix} p\left\{\begin{pmatrix} \overbrace{1 \ \dots \ 1}^{p} & \overbrace{0 \ \dots \ 0}^{p} & 0 \ \dots \ 0 \\ 0 \ \dots \ 0 & 1 \ \dots \ 1 & 0 \ \dots \ 0 \\ 0 \ \dots \ 0 & 0 \ \dots \ 0 & \star \ \dots \ \star \\ \vdots \quad\quad \vdots & \vdots \quad\quad \vdots & \vdots \quad\quad \vdots \\ 0 \ \dots \ 0 & 0 \ \dots \ 0 & \star \ \dots \ \star \end{pmatrix}\right. \\ \begin{matrix} \star \ \dots \ \star & \star \ \dots \ \star & \star \ \dots \ \star \\ \vdots \quad\quad \vdots & \vdots \quad\quad \vdots & \vdots \quad\quad \vdots \\ \star \ \dots \ \star & \star \ \dots \ \star & \star \ \dots \ \star \end{matrix} \end{pmatrix}.$$

By applying the same arguments on the 3^{rd} to the p^{th} row of PAP^T, we can update P such that

$$PAP^T = \begin{pmatrix} I_p \otimes \mathbf{1}_p^T & 0 & \dots & 0 \\ \star & \star & \dots & \star \\ \vdots & \vdots & & \vdots \\ \star & \star & \dots & \star \end{pmatrix}.$$

From this, we deduce that the first row of PA^2P^T is then $(\mathbf{1}_{p^2}^T\ 0 \dots 0)$. Moreover, since $PA^3P^T = (PA^2P^T)(PAP^T)$ is binary, we deduce that there is at most one nonzero entry among the first p^2 entries of each column. Therefore, we can update P so that

$$PAP^T = \begin{pmatrix} p \text{ blocks of } p \text{ rows} \left\{ \begin{pmatrix} I_p \otimes \mathbf{1}_p^T & 0 & \dots & \dots & 0 \\ 0 & I_p \otimes \mathbf{1}_p^T & 0 & \dots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & I_p \otimes 1_p^T & 0 \end{pmatrix} \right. \\ \begin{matrix} \star & \dots & \dots & \dots & \dots & \dots & \star \\ \vdots & & & & & & \vdots \\ \star & \dots & \dots & \dots & \dots & \dots & \star \end{matrix} \end{pmatrix}.$$

By repeating this $m-1$ times, we prove that we can update P so that

$$PAP^T = \begin{pmatrix} n/p \text{ rows} \left\{ \begin{matrix} \overbrace{1 \quad \dots \quad 1}^{p} & & & \\ & \overbrace{1 \quad \dots \quad 1}^{p} & & \\ & & \ddots & \\ & & & \overbrace{1 \quad \dots \quad 1}^{p} \end{matrix} \right. \\ \vdots \end{pmatrix}.$$

In addition, a simple induction on $1 \leq k \leq m$ shows that

$$(PAP^T)^k = \begin{pmatrix} I_{n/p^k} \otimes \mathbf{1}_{p^k}^T \\ \vdots \end{pmatrix}.$$

Since A is p-regular with rank n/p, the rest of the above matrix PAP^T is made up of rows chosen among the first n/p ones. We write the matrix PAP^T

as

$$PAP^T = \begin{pmatrix} B_1 \\ \vdots \\ B_p \end{pmatrix}$$

where each block B_i has n/p rows.

Up to a row permutation, all these blocks B_i are identical. Indeed, if it was not the case, a block B_i would have two identical rows. Hence, there would be a column such that in B_i, the sum of the entries in that column is greater than 1. However, since

$$(PAP^T)^{m-1} = \begin{pmatrix} I_p \otimes \mathbf{1}_{n/p}^T \\ \vdots \end{pmatrix},$$

$(PAP^T)^m = (PAP^T)^{m-1}(PAP^T)$ would not be a binary matrix, which is a contradiction since $(PAP^T)^m = J_n$.

Therefore, we have shown that there is a permutation matrix P such that

$$PAP^T = \begin{pmatrix} I_{n/p} \\ Q_2 \\ \vdots \\ Q_p \end{pmatrix} \otimes \mathbf{1}_p^T,$$

where each $Q_i \in \{0,1\}^{(n/p)\times(n/p)}$ is a permutation matrix. □

Recall that given matrices $Q_1, ..., Q_s \in \mathbb{R}^{n\times n}$, the block diagonal matrix where the diagonal blocks are $Q_1, ..., Q_s$ is denoted by $\mathrm{diag}(Q_1, ..., Q_s)$.

Theorem 5.14. *Let $A \in \{0,1\}^{n\times n}$ be p-regular. If $A^m = J_n$ and $\mathrm{rank}(A) = n/p$, then A is isomorphic to a matrix*

$$PD(p,n),$$

where $P = \mathrm{diag}(Q_1, ..., Q_p)$ and each $Q_i \in \{0,1\}^{(n/p)\times(n/p)}$ is a permutation matrix.

Proof. We have seen in the previous lemma that A is isomorphic to a matrix of the form

$$\begin{pmatrix} I_{n/p} \\ Q_2 \\ \vdots \\ Q_p \end{pmatrix} \otimes \mathbf{1}_p^T,$$

where each $Q_i \in \{0,1\}^{(n/p)\times(n/p)}$ is a permutation matrix. Such a matrix can be written as

$$P(\mathbf{1}_p \otimes I_{n/p}) \otimes \mathbf{1}_p^T,$$

with $P = \text{diag}(I_{n/p}, Q_2, ..., Q_p)$. Hence, A is isomorphic to $PD(p,n)$. □

Of course, not all the matrices of the form $PD(p,n)$ like in the previous theorem are solutions to $A^m = J_n$. Indeed, the matrix in Remark 5.12 is an example of such a matrix which is not a solution. Therefore, the previous result is not a full characterization of the solutions with minimum rank.

In 1967, Hoffman [45] was interested in a characterization of the binary solutions to the equation $A^2 = J_n$. Characterizing these solutions is still an open problem and to our best knowledge, no subclass of solutions has been characterized. From Theorem 5.14, we provide a characterization of the solutions to $A^2 = J_n$ with minimum rank.

Corollary 5.15. *Let $A \in \{0,1\}^{n\times n}$ be a p-regular solution to $A^2 = J_n$ with minimum rank. Then, A is isomorphic to the De Bruijn matrix $D(p,n)$.*

Proof. Lemma 5.1 states that $n = p^2$. Moreover, from Theorem 5.14, we know that

$$A \cong PD(p,n)$$

where $P = \text{diag}(Q_1, ..., Q_p)$ and each $Q_i \in \{0,1\}^{p\times p}$ is a permutation matrix.

However, because of the structure of $D(p,n)$, it follows that

$$PD(p,n) = PD(p,n)P^T.$$

As a consequence, $A \cong D(p,n)$. □

Since we already know that $D(p,n)$ with $n = p^2$ is a solution to $A^2 = J_n$ with minimum rank, we deduce from this and Corollary 5.15 a characterization of the binary solutions to $A^2 = J_n$ with minimum rank. This characterization is written in the following theorem.

Theorem 5.16. *Let $A \in \{0,1\}^{n\times n}$ be a p-regular matrix. Then, A is a solution to $A^2 = J_n$ with minimum rank if and only if A is isomorphic to $D(p,n)$.*

As shown in Remark 5.8, not all the solutions to $A^2 = J_n$ have minimum rank.

5.4 A class of roots of J_n isomorphic to a De Bruijn matrix

In [101], it is proved that some g-circulant binary roots of J_n are isomorphic to a De Bruijn matrix. More specifically, the following result is shown.

Proposition 5.17. *Let A be a g-circulant binary solution to $A^m = J_n$ which is p-regular. If $g^m = 0 \mod n$, then A is isomorphic to the De Bruijn matrix $D(p,n)$.*

In this section, we complete the results of [101] by identifying another class of binary solutions to $A^m = J_n$ isomorphic to a De Bruijn matrix.

Definition 5.18. *A nice permutation matrix is built as follows: start with a $p \times p$ permutation matrix. Then, replace all the zeros by a $p \times p$ zero matrix and each one by a $p \times p$ permutation matrix. Repeat this m times. We obtain a permutation matrix of dimension p^m called nice permutation matrix.*

Observation: let A be an $n \times n$ matrix such that $n = p^m$ for some integers p and m. Then, A is made up of p row-blocks of n/p rows each: the first row-block contains the first n/p rows of A, the second row-block contains the next n/p rows of A, etc. In the same way, each of these row-blocks is made up of p row-blocks of n/p^2 rows. Recursively, we decompose each block in this way until we have row-blocks of one row. From the definition of a nice permutation matrix, it is straightforward that multiplying A to the left by a nice permutation matrix performs in each row-block of n/p^{i-1} rows, a permutation of the p row-blocks of n/p^i rows.

As an example, take $p = m = 2$,

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix}$$

and the nice permutation

$$P = \begin{pmatrix} & & 1 & 0 \\ & & 0 & 1 \\ 0 & 1 & & \\ 1 & 0 & & \end{pmatrix}.$$

Then,

$$PA = \begin{pmatrix} 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \\ 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 \end{pmatrix}.$$

All this section is based on this effect of a nice permutation matrix when it multiplies a matrix to the left.

Definition 5.19. *A matrix of the form $PD(p,n)$, where P is a nice permutation matrix, is called a nice permutation of the De Bruijn matrix $D(p,n)$.*

The goal of this section is to prove Corollary 5.24, stating that a nice permutation of a De Bruijn matrix $D(p,n)$ (with $n = p^m$) is an m^{th} root of J_n isomorphic to $D(p,n)$.

Definition 5.20. *Let $P \in \{0,1\}^{n\times n}$ with $n = p^m$. The matrix P is called a nice permutation matrix of level i ($1 \leq i \leq m$) if it is a nice permutation matrix performing only permutations of row-blocks of p^{i-1} rows.*

For instance, if $p = 2$ and $m = 3$, the matrix

$$P = \begin{pmatrix} & & 1 & 0 & & & & \\ & & 0 & 1 & & & & \\ 1 & 0 & & & & & & \\ 0 & 1 & & & & & & \\ & & & & & & 1 & 0 \\ & & & & & & 0 & 1 \\ & & & & 1 & 0 & & \\ & & & & 0 & 1 & & \end{pmatrix}$$

is a nice permutation of level 2.

The identity matrix I_{p^m} is considered as a nice permutation matrix of level i, for each $1 \leq i \leq m$.

It should be noted that if P is a nice permutation of level i, then P^T is also a nice permutation of level i.

Lemma 5.21. *Let $\tilde{D}(p,n)$ be a nice permutation of the De Bruijn matrix $D(p,n)$. Then, for each nice permutation matrix P_i of level i, there are nice permutation matrices $P_{i_1}, \ldots, P_{i_s}$ of level less than i such that*

$$\tilde{D}(p,n)P_i^T = P_{i_1} \ldots P_{i_s} \tilde{D}(p,n).$$

Proof. The matrix $\tilde{D}(p,n)$ is made up of p column-blocks of n/p columns: the first column-block contains the first n/p columns of $\tilde{D}(p,n)$, the second one contains the next n/p columns, etc. Each of them is similarly made up of p column-blocks of n/p^2 columns, and recursively until we get column-blocks of 1 column.

As observed at the beginning of this section, multiplying $\tilde{D}(p,n)$ to the right by P_i^T means permuting in each column-block of p^i columns, the p column-blocks with p^{i-1} columns.

Due to the structure of $D(p,n)$, if $i=1$, then $\tilde{D}(p,n)P_i^T = \tilde{D}(p,n)$ and the result is clear. Suppose now $1 < i \leq m$.

It follows from the structure of $D(p,n)$ that in $\tilde{D}(p,n)$, the nonzero entries of each column-block of p^{i-1} (resp. p^i) columns are the ones of the rows in row-blocks with p^{i-2} (resp. p^{i-1}) rows.

As a consequence, permuting the column-blocks with p^{i-1} columns inside a column-block of p^i columns is equivalent to permuting inside each row-block of p^{i-1} rows, the row-blocks with p^{i-2} rows and maybe also to performing some nice permutations inside the row-blocks with p^{i-2} rows.

All these permutations are therefore nice permutations of level less than i.

□

As an example, consider the following nice permutation matrix of the De Bruijn matrix $D(2,8)$:

$$\tilde{D}(2,8) = \begin{pmatrix} 1 & 1 & & & & & & \\ & & 1 & 1 & & & & \\ & & & & & & 1 & 1 \\ & & & & 1 & 1 & & \\ 1 & 1 & & & & & & \\ & & 1 & 1 & & & & \\ & & & & 1 & 1 & & \\ & & & & & & 1 & 1 \end{pmatrix}.$$

If P_3 is the nice permutation matrix of level 3 that permutes the two blocks of

4 rows, we have:

$$\tilde{D}(2,8)P_3^T = \begin{pmatrix} & & & & 1 & 1 & & \\ & & & & & & 1 & 1 \\ & & 1 & 1 & & & & \\ 1 & 1 & & & & & & \\ & & & & 1 & 1 & & \\ & & & & & & 1 & 1 \\ 1 & 1 & & & & & & \\ & & 1 & 1 & & & & \end{pmatrix}.$$

Notice that if we take these nice permutation matrices P_1 and P_2 of level 1 and 2 respectively:

$$P_1 = \begin{pmatrix} 0 & 1 & & & & & & \\ 1 & 0 & & & & & & \\ & & 0 & 1 & & & & \\ & & 1 & 0 & & & & \\ & & & & 1 & 0 & & \\ & & & & 0 & 1 & & \\ & & & & & & 1 & 0 \\ & & & & & & 0 & 1 \end{pmatrix} \quad P_2 = \begin{pmatrix} & & 1 & 0 & & & & \\ & & 0 & 1 & & & & \\ 1 & 0 & & & & & & \\ 0 & 1 & & & & & & \\ & & & & & & 1 & 0 \\ & & & & & & 0 & 1 \\ & & & & 1 & 0 & & \\ & & & & 0 & 1 & & \end{pmatrix},$$

then $P_2P_1\tilde{D}(2,8) = \tilde{D}(2,8)P_3^T$.

The following lemma will be useful to prove that a nice permutation of the De Bruijn matrix $D(p,n)$ is isomorphic to $D(p,n)$.

Lemma 5.22. *Let $\tilde{D}(p,n)$ be a nice permutation of the De Bruijn matrix $D(p,n)$ (with $n = p^m$) and P_i be a nice permutation matrix of level i. Then, $P_i\tilde{D}(p,n)$ is isomorphic to $\tilde{D}(p,n)$.*

Proof. The proof is by induction on the level i.

- If $i = 1$, it is clear that $P_i\tilde{D}(p,n) = P_i\tilde{D}(p,n)P_i^T$.

- If $i > 1$, Lemma 5.21 claims that multiplying to the right $P_i\tilde{D}(p,n)$ by P_i^T is equivalent to performing nice permutations of level less than i on the rows of $P_i\tilde{D}(p,n)$.

Hence, there is a nice permutation matrix $\tilde{P}$ performing only permutations of

level less than i such that

$$\tilde{P}P_i\tilde{D}(p,n) = P_i\tilde{D}(p,n)P_i^T.$$

Therefore, $P_i\tilde{D}(p,n)P_i^T$ is a nice permutation of $D(p,n)$. Since $\tilde{P}^T$ is a product of nice permutation matrices of level less than i, by induction, we know that $\tilde{P}^T P_i\tilde{D}(p,n)P_i^T$ is isomorphic to $P_i\tilde{D}(p,n)P_i^T$ and therefore isomorphic to $\tilde{D}(p,n)$.

As a consequence, since $P_i\tilde{D}(p,n) = \tilde{P}^T P_i\tilde{D}(p,n)P_i^T$, $P_i\tilde{D}(p,n)$ is isomorphic to $\tilde{D}(p,n)$.

□

Proposition 5.23. *A nice permutation of the De Bruijn matrix $D(p,n)$ is isomorphic to $D(p,n)$.*

Proof. A nice permutation of $D(p,n)$ can be written as $P_m...P_2P_1D(p,n)$, where each P_i is a nice permutation matrix of level i.

By repeatedly applying the previous lemma, it follows that such a matrix is isomorphic to $D(p,n)$. □

Corollary 5.24. *A nice permutation of the De Bruijn matrix $D(p,n)$ ($n = p^m$) is a m^{th} root of J_n, isomorphic to $D(p,n)$.*

Proof. This follows from Proposition 5.23 and the fact that $D(p,n)^m = J_n$. □

The example in Remark 5.9 shows that not each root of J_n with minimum rank is a nice permutation of a De Bruijn matrix.

5.5 Conclusion

In 1967, Hoffman [45] was interested in a characterization of the binary solutions to the equation $A^2 = J_n$. Since then, much research has been done about the binary solutions not only of $A^2 = J_n$, but also of the more general equations $A^m = J_n$. However, despite the attention this topic has received, very few results have been discovered.

The De Bruijn matrices are the only known solutions of $A^m = J_n$. Nevertheless, since each binary solution to $A^m = J_n$ has been proved to share several

properties with the De Bruijn matrices, we should investigate if there is a relation between a general solution to $A^m = J_n$ and the De Bruijn matrices.

The only known result in this direction is that some circulant solutions to $A^m = J_n$ are isomorphic to a De Bruijn matrix [101]. In this chapter, we have presented further results about the connection between the De Bruijn matrices and a general solution to $A^m = J_n$.

We have proved that each binary solution to $A^m = J_n$ with minimum rank is isomorphic to a row permutation of a De Bruijn matrix. Moreover, we have characterized the solutions to $A^2 = J_n$ with minimum rank through the De Bruijn matrices. This latter result partially solves Hoffman's open problem of characterizing the binary solutions to $A^2 = J_n$. Finally, we have identified a class of solutions to $A^m = J_n$ isomorphic to a De Bruijn matrix. All these results were published in [96]:

M. Trefois, P. Van Dooren and J.-C. Delvenne, *Binary factorizations of the matrix of all ones*, Linear Algebra and Its Applications, 468: 63-79, 2015.

Despite our results, we are far from understanding the solutions to $A^m = J_n$. Further research about this topic should be done. Here are some questions it would be interesting to answer:

- Among the matrices that are a row permutation of the De Bruijn matrix $D(p,n)$ (with $n = p^m$), can we identify the ones that solve $A^m = J_n$?
- Can we characterize the binary solutions to $A^m = J_n$ with minimum rank ?
- Can each binary solution to $A^m = J_n$ be expressed from De Bruijn matrices ?

Chapter 6

Linear solvers with fast iterations

Even for highly structured linear systems, the iterative algorithms, such as gradient descent, used to approximate the minimal norm solution of the system have a computational complexity per iteration that is at least linear in the system size.

During these last ten years, the hope has been to develop new iterative methods in order to solve a linear system in nearly-linear time. Recall that by *nearly-linear time* we mean a computational complexity of the form:

$$\mathcal{O}(m \log^c m),$$

where m denotes the size of the system or the number of nonzero entries in the system matrix, and c is an arbitrary constant.

In order to contribute to the problem of finding nearly-linear time solvers, we provide in this chapter a new iterative algorithm with fast iterations, that approximates the minimal norm solution of the system. Specifically, we define a new matrix factorization called k-sparse and based on this factorization, we present a new iterative method for under-determined systems where the running time per iteration is $\mathcal{O}(k)$.

We warn the reader that our results focus on the computational complexity of an iteration. The problem of determining the number of needed iterations is not considered. Moreover, the present chapter only contains theoretical results. We do not compare our method with existing ones by numerical experiments.

In Section 6.1, we define a k-sparse matrix factorization and show how to compute in $\mathcal{O}(k)$ time an iteration of the form

$$x_{t+1} = x_t - \frac{x_t^T q}{q^T q} \cdot q,$$

where q is a column of a k-sparsely factorizable matrix.

In Section 6.2, we present the hierarchical matrices, defined in [42], as an example of k-sparsely factorizable matrices where k depends on the depth and the degree of the hierarchical structure, but not directly on the size of the matrix.

In Section 6.3, we present our iterative algorithm that computes the minimal norm solution of an under-dertermined linear system and apply it when the system matrix is the incidence matrix of a graph.

Section 6.4 showcases how we use our algorithm to solve in nearly-linear time a Laplacian system. Precisely, the goal is to find in nearly-linear time an approximate solution of a compatible system $Ly = b$, where L is the Laplacian matrix of a connected undirected graph. To do so, we factorize L as $L = BB^T$, where B denotes an incidence matrix of the graph. Then, we set $x = B^T y$ and we use our algorithm to find in nearly-linear time the minimal norm solution $x^\star$ to $Bx = b$. Then, given $x^\star$, we come back to the original coordinates y by solving in linear time an almost triangular system. By a proof similar to the one in [56], it can be shown that the resulting vector $y^\star$ is a good approximate solution to $Ly = b$.

Finally, Section 6.5 concludes the chapter and discusses possible avenues for future work.

The results we present in this chapter were submitted as a journal paper in 2016 [95]:

M. Trefois, M.T. Schaub, J.-C. Delvenne and P. Van Dooren, *A new sparse matrix factorization for linear solvers with fast iterations*, submitted, 2016.

6.1 Sparse matrix factorization and computational complexity

Let x_t and q be two vectors of $\mathbb{R}^m$ and consider the following iteration*:

$$x_{t+1} = x_t + \frac{x_t^T q}{q^T q} \cdot q. \tag{6.1}$$

If x_{t+1} was naively computed, then the cost of this iteration would be linear in m. In this section, we define the notion of k-sparse matrix factorization and

*All the results of this chapter also hold if the scalar product is of the form $\langle x_t, q\rangle_R = x_t^T Rq$ where $R \in \mathbb{R}^{m\times m}$ is a positive definite diagonal matrix. But for the sake of simplicity, we choose R as being the identity matrix.

show that if q is a column of a k-sparsely factorizable matrix, then the iteration (6.1) can be computed in $\mathcal{O}(k)$ time. This will be the key ingredient of our results on linear solvers presented in Section 6.3.

Definition 6.1. *The support of a vector $v = (v_1, \dots, v_m)^T \in \mathbb{R}^m$ is set of indices of the nonzero entries of v:*

$$\operatorname{supp}(v) = \{i \in \{1, \dots, m\} : v_i \neq 0\}.$$

A vector $v \in \mathbb{R}^m$ is said to be k-sparse, if the size of its support, $|\operatorname{supp}(v)|$, is less than or equal to k. Similarly, a matrix is said to be k-column (k-row) sparse if each of its columns (rows) is k-sparse.

While performing the iterations (6.1), suppose that x_t is stored, not in the canonical base, but as a linear combination of the columns of C, i.e. as a vector y_t such that $x_t = Cy_t$. Then computing the next iterate requires to compute the scalar product $x_t^T q$, which turns out to be the most difficult step of the iteration. It is computed as $y_t^T C^T C D_i$ for some column D_i of D. To cap the complexity of this operation, one must understand the sparsity pattern of $C^T C$, ruled by the overlap in support of columns of C. If every column of C overlaps in support with at most c other columns, then every column of $C^T C$ contains at most c non-zero entries. If moreover every column D_i contains at most c' entries, then $C^T C D_i$ is a cc'-sparse vector, and $y_t C^T C D_i$ is computed in time $\mathcal{O}(k)$, for $k = cc'$. We say therefore that the factorization $Q = CD$ is k-sparse. In fact this definition can and must be improved to reach tighter complexity bounds. First, we can exploit (non-trivially) the symmetry of $C^T C$, decomposable as $U^T + U$. The number of non-zero entries in the ith column of U^T (or ith row of U) is bounded by the number of columns c_j that overlaps with c_i for $j \geq i$. Second, two columns of U^T may have their non-zero entries placed in the same positions, therefore a sum of them will still have the same or smaller support. What matters for the complexity in the end is therefore the cardinality of the union of supports of all columns j of U^T, where j belongs to the support of D_i—possibly much lower than the rough estimate cc'. This justifies the following definition.

Definition 6.2. *Suppose a matrix $Q \in \mathbb{R}^{m \times n}$ has a factorization $Q = CD$. Let us denote the columns of $C \in \mathbb{R}^{m \times p}$ and $D \in \mathbb{R}^{p \times n}$ by c_i and d_j, respectively. We define the forward-overlap $FO(c_i)$ of a column c_i to be the list of columns c_j, with $j \geq i$, that have a support overlapping with the support of c_i. We call the factorization $Q = CD$ k-sparse if $\left|\cup_{i \in \operatorname{supp}(d_j)} FO(c_i)\right| \leq k$ for all j. Note that without loss of generality each column of C and each row of D is supposed to be nonzero.*

The example in Figure 6.2 shows a 16-sparse[†] factorization of the given matrix E. Albeit this factorization is obtained from a general technique explained in the next section, one can already check for instance that the forward overlap of column c_{12} is $FO(c_{12}) = \{c_{12}, c_{13}, c_{16}, c_{20}\}$ and $\left|\cup_{i\in \mathrm{supp}(d_5)} FO(c_i)\right| = \left|\{c_{11}, c_{12}, c_{13}, c_{15}, c_{16}, c_{19}, c_{20}\}\right| = 7 \leq 16$.

Here are some straightforward properties of a k-sparse factorization.

1) If $Q = CD$ is a k-sparse factorization, then for each column c_i of C, $|FO(c_i)| \leq k$, C is k-row sparse and each column of D is k-sparse[‡].

2) If $Q = CD$ is a k-sparse factorization and F is f-column sparse, then $QF = C(DF)$ is a kf-sparse factorization of QF.

3) If $Q_1 = C_1D_1$ is a k_1-sparse factorization and $Q_2 = C_2D_2$ is a k_2-sparse factorization, then the matrix $\begin{pmatrix} Q_1 \\ Q_2 \end{pmatrix}$ is $(k_1 + k_2)$-sparsely factorizable. In order see this, we write

$$\begin{pmatrix} Q_1 \\ Q_2 \end{pmatrix} = \begin{pmatrix} C_1 & \\ & C_2 \end{pmatrix} \begin{pmatrix} D_1 \\ D_2 \end{pmatrix}.$$

The following theorem gives the running time of N iterations of the form (6.1) when the vectors q are the columns of a k-sparsely factorizable matrix.

Theorem 6.3. *Let $Q \in \mathbb{R}^{m\times n}, C \in \mathbb{R}^{m\times p}$ and $D \in \mathbb{R}^{p\times n}$ be matrices such that $Q = CD$ is a k-sparse factorization of Q. Then, the computational complexity of N iterations of the form (6.1) where q is a column of Q and that start from an arbitrary vector $x_0 \in \mathbb{R}^m$ is:*

$$\mathcal{O}(Nk + (p+m)k + nk^2 + Cost(C^TC)),$$

where $Cost(C^TC)$ is the cost of computing C^TC.

Proof. At first, we show that we can assume without loss of generality that the columns of C contain the canonical basis of $\mathbb{R}^m$. Indeed, set $\tilde{C} := \begin{pmatrix} I_m & C \end{pmatrix} \in \mathbb{R}^{m\times(p+m)}$ and $\tilde{D} := \begin{pmatrix} 0 \\ D \end{pmatrix} \in \mathbb{R}^{(p+m)\times n}$. It follows that for each column $\tilde{c}_i$ of $\tilde{C}$,

[†]The proof of Theorem 6.9 ensures that $E = CD$ is a k-sparse factorization for $k = 16$. However, in this example, we compute by hand that the value of k could even be reduced to 8.

[‡]It should be noted that this property holds because in the definition of sparse factorization each column of C and each row of D is asked to be nonzero. This restriction on the columns of C and the rows of D is indeed required in order to have this first property.

$|FO(\tilde{c}_i)| \leq k+1$, that $\tilde{D}$ is $(k+1)$-column sparse and that for each column $\tilde{d}_j$, $\left|\cup_{i\in \mathrm{supp}(\tilde{d}_j)} FO(\tilde{c}_i)\right| \leq k+1$. Consequently, even though $\tilde{D}$ has some zero rows, the factorization $\tilde{C}\tilde{D}$ has all the properties of a $(k+1)$-sparse factorization and we say that $\tilde{C}\tilde{D}$ is $(k+1)$-sparse. Moreover,

$$\tilde{C}^T\tilde{C} = \begin{pmatrix} I & C \\ C^T & C^TC \end{pmatrix}$$

and the complexity of computing $\tilde{C}^T\tilde{C}$ is given by the complexity of computing C^TC. As a consequence, following the complexity given in the theorem, the running time does not depend on the choice of the decomposition CD or $\tilde{C}\tilde{D}$.

Consequently, we can assume without loss of generality that a vector $h_0 \in \mathbb{R}^p$ such that $x_0 = Ch_0$ can be computed in $\mathcal{O}(m)$ time.

Denote by U the $p \times p$ upper triangular matrix such that $C^TC = U^T + U$. Notice that the i^{th} row of U is $|FO(c_i)|$-sparse. In particular, since $|FO(c_i)| \leq k$, we can also say that the matrix U is k-row sparse. Moreover, since each column of C is assumed to be nonzero, it should be noticed that U is invertible.

In order to get a running time for each iteration not depending on m, we use these properties on U and we deduce two generating sets of $\mathbb{R}^m$ in which each column of Q has a decomposition with a sparsity depending only on k. If h_i and k_i are a decomposition of x_i in these generating sets, then we show that the cost of the iterations on h_i and k_i only depends on k. These generating sets are both needed to be able to compute each scalar product $x_i^Tq_j$, where q_j is a column of Q, in a running time depending only on k.

We show below that the columns of C and CU^{-T} are two appropriate generating sets. Note that if $x = Ch$, then a vector k such that $x = CU^{-T}k$ is given by $k = U^Th$.

a) Given $x_0 = Ch_0$, since U^T is k-column sparse, we compute the vector $k_0 := U^Th_0$ in time $\mathcal{O}(pk)$.

b) Let d_j be a column of D, which is k-sparse. Then, since $Q = CD$ is a k-sparse factorization, the vector $e_j := U^Td_j$ is k-sparse and is computed in time $\mathcal{O}(k^2)$.

c) Given a vector $x_i \in \mathbb{R}^m$ with $h_i, k_i \in \mathbb{R}^p$ such that $x_i = Ch_i$ and $k_i = U^Th_i$ and given a column $q_j = Cd_j$ of Q with $e_j := U^Td_j$, we compute $x_i^Tq_j$ in

time $\mathcal{O}(k)$. Indeed, we compute that

$$\begin{aligned} x_i^T q_j &= x_i^T q_j \\ &= h_i^T (C^T C) d_j \\ &= h_i^T (U + U^T) d_j \\ &= (U^T h_i)^T d_j + h_i^T (U^T d_j) \\ &= k_i^T d_j + h_i^T e_j. \end{aligned}$$

Since e_j and d_j are k-sparse (see b)), it takes $\mathcal{O}(k)$ time to compute $x_i^T q_j$.

d) Given $x_i \in \mathbb{R}^m$ and a column $q_j = Cd_j$ of Q, the next iteration is

$$x_{i+1} = x_i - \frac{x_i^T q_j}{q_j^T q_j} \cdot q_j.$$

Given $h_i, k_i \in \mathbb{R}^p$ such that $x_i = Ch_i$ and $k_i = U^T h_i$ and given $e_j = U^T d_j$, the vectors

$$h_{i+1} := h_i - \frac{x_i^T q_j}{q_j^T q_j} \cdot d_j$$

$$k_{i+1} = k_i - \frac{x_i^T q_j}{q_j^T q_j} \cdot e_j$$

are such that $x_{i+1} = Ch_{i+1}$ and $k_{i+1} = U^T h_{i+1}$. Moreover, from b), c) and the fact that d_j is k-sparse, given h_i, k_i, d_j and e_j, the vectors h_{i+1} and k_{i+1} are computed in time $\mathcal{O}(k)$.

e) Given $h_N \in \mathbb{R}^p$ such that $x_N = Ch_N$, we compute x_N in time $\mathcal{O}(mk)$. Indeed, this is due to the fact that C is k-row sparse.

Consequently, at each iteration, we only use the vectors h_i, k_i, d_j and e_j in order to compute h_{i+1} and k_{i+1}. The vectors h_{i+1} and k_{i+1} are both needed to compute $x_{i+1}^T q_j$ in time $\mathcal{O}(k)$. Finally, the approximation x_N is computed from the relation $x_N = Ch_N$.

It follows from the previous observations that the method runs in time

$$\mathcal{O}(Nk + (p+m)k + nk^2 + Cost(C^T C)).$$

Indeed, we compute

1) the matrix U for which the cost is given by $Cost(C^T C)$

2) $h_0 \in \mathbb{R}^p$ such that $x_0 = Ch_0$ in time $\mathcal{O}(m)$

3) $k_0 := U^T h_0$ in time $\mathcal{O}(pk)$ (see a)),

4) $U^T D$ in time $\mathcal{O}(nk^2)$ (see b))

5) all the scalar products $q^T q$ (q being a column of Q) in time $\mathcal{O}(nk)$ (see c)),

6) each iteration on h_i and k_i in time $\mathcal{O}(k)$ (see d)) and

7) $x_N = Ch_N$ in time $\mathcal{O}(mk)$ (see e)).

□

Since the size of the forward-overlap of each column of C is at most k, we deduce that the cost of computing $C^T C$ is given by $\mathcal{O}(pmk)$. However, we will see in the next sections that this cost could be reduced in some particular cases.

The remarkable point about Theorem 6.3 is that it shows that the running time of *each iteration* can be made *independent* of m. More precisely, the cost of each iteration is merely $\mathcal{O}(k)$. Recall from our discussion above that if each iteration was performed naively, we would deduce from (1.1) that the cost of each iteration is linear in m. Since k is (normally) much smaller than m, this means the cost per iteration can be largely reduced due to the k-sparse factorization.

We can use Theorem 6.3, for instance, in order to efficiently approximate the minimal norm solution of under-determined linear systems.

6.2 An important class of k-sparsely factorizable matrices: Hierarchical matrices

As our above result illustrates, k-sparsely factorizable matrices are extremely powerful to reduce the complexity of an iteration of the form (6.1). A natural question is therefore what kind of matrices are k-sparsely factorizable. In the following, we will discuss hierarchical $\mathscr{H}_r$-matrices and show that they are k-sparsely factorizable where k depends only on the depth and the degree of the hierarchical structure. Applications of this result to incidence matrices and Laplacian systems are discussed in the sequel.

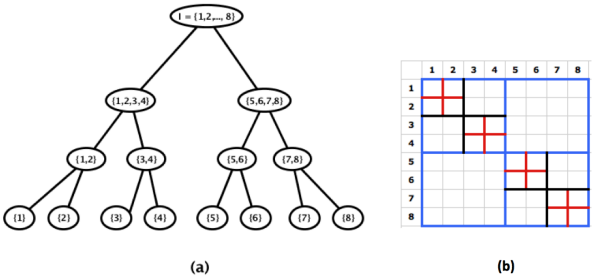


Figure 6.1: (a) A dendrogram of $I = \{1, ..., 8\}$. (b) The $\mathscr{P}$-partitioning of an 8×8 matrix where $\mathscr{P}$ is the dendrogram of $\{1, ..., 8\}$ in (a).

6.2.1 Definition of an $\mathscr{H}_r$-matrix

As the name suggests, $\mathscr{H}_r$-matrices are intimately related to hierarchical structures. As a hierarchy may be aptly represented as a tree we introduce these matrices here with the help of trees.

Definition 6.4. *A* dendrogram *is a hierarchical partitioning $\mathscr{P}$ of the set $\{1, \ldots, n\}$, namely a sequence of increasingly finer partitions $P_0, \ldots, P_h$ starting from the coarsest (global) partition P_0 given by the whole set, up to the finest (singleton) partition P_h into n sets. This dendrogram is conveniently represented by a rooted directed tree. The nodes of this tree at depth i are the subsets of partition P_i. Thus the root ($i = 0$) is the full set while the leaves ($i = h$) are the n single-element subsets. The children (out-neighbours) of a node at depth i correspond to the subsets of this node as specified by the next partition P_{i+1}. We call h the* depth *of the dendrogram, and its* maximum degree *the maximum number of children of a node in the tree.*

As an example, Figure 6.1(a) shows a dendrogram of $\{1, ..., 8\}$, with depth 3 and maximum degree 2. For simplicity of notation, we suppose throughout the chapter that every node of a dendrogram has consecutive elements.

A dendrogram $\mathscr{P}$ induces a hierarchical block segmentation of a matrix $E \in \mathbb{R}^{n \times n}$ as follows. The rows and columns of E are first block-partitioned according to the partition P_1:

$$E = \begin{pmatrix} E_{I_1 \times I_1} & E_{I_1 \times I_2} & \cdots & E_{I_1 \times I_t} \\ E_{I_2 \times I_1} & E_{I_2 \times I_2} & \cdots & E_{I_2 \times I_t} \\ \vdots & \vdots & \vdots & \vdots \\ E_{I_t \times I_1} & E_{I_t \times I_2} & \cdots & E_{I_t \times I_t} \end{pmatrix}, \tag{6.2}$$

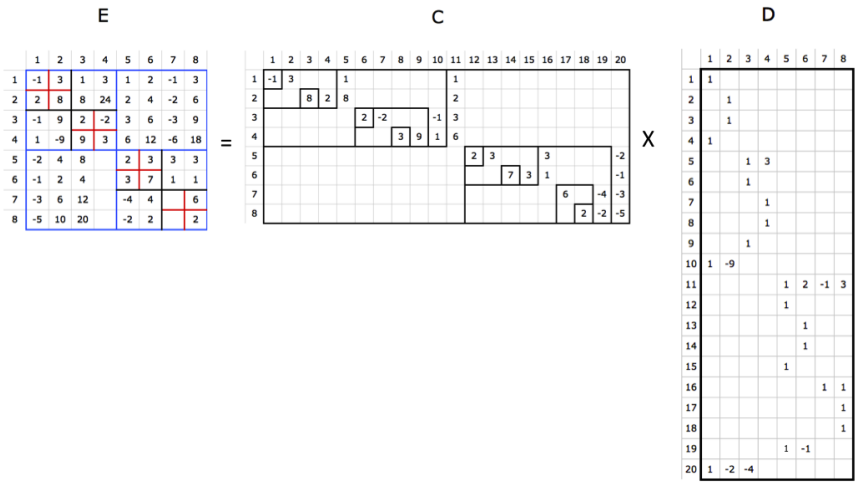


Figure 6.2: The matrix E is an $\mathscr{H}_1(\mathscr{P})$-matrix with $\mathscr{P}$ the dendrogram in Figure 6.1(a). The factorization $E = CD$ is 16-sparse according to the proof of Theorem 6.9. However, we compute by hand that $E = CD$ is also a 8-sparse factorization.

where $I_1, \ldots, I_t$ are the elements of partition P_1 , and we assume for simplicity of notation sets I_i of consecutive integers. The diagonal blocks $E_{I_i \times I_i}$, are recursively sub-partitioned according to P_2, etc. This partitioning of E is called $\mathscr{P}$*-partitioning*. See Figure 6.1(b) for an illustration.

Definition 6.5. *We use the term* elementary block *to refer to a sub-matrix $E_{I_i \times I_j}$ of E (see Eq.* (6.2)*), where I_i and I_j are sets appearing in some partition P_k, that is NOT sub-divided by a partition finer than P_k.*

Definition 6.6. *An $\mathscr{H}_r(\mathscr{P})$-matrix is a square matrix, that is structured according to the dendrogram $\mathscr{P}$ and where the elementary blocks have rank at most $r \in \mathbb{N}$. We use the shorthand $\mathscr{H}_r$ when the dendrogram is clear from context.*

It should be noticed that a sub-matrix $E_{I_i \times I_i}$ of an $\mathscr{H}_r(\mathscr{P})$-matrix E, where I_i is a set of some partition P_k, is an $\mathscr{H}_r$-matrix as well.

6.2.2 Sparse factorization property

In what follows, we show that an $\mathscr{H}_r(\mathscr{P})$-matrix is k-sparsely factorizable and express k in terms of r, d and h with h the depth of the dendrogram $\mathscr{P}$ and d its maximum degree.

Recall that an $\mathscr{H}_r(\mathscr{P})$-matrix E is of the form (6.2).From this hierarchical structure of E, we could imagine to build a matrix C from a basis of the column space of the elementary blocks. If C is constructed properly, it should be easy to find a matrix D such that $E = CD$ is a k-sparse factorization. We denote by B_{ij} a matrix in which the columns are a basis of the column space of the elementary block $E_{I_i \times I_j}$. Since the elementary blocks are of rank at most r, the matrices B_{ij} have at most r columns. When $d = 3$, we recursively define the matrix C as:

$$C = \begin{pmatrix} C_1 & B_{12} & B_{13} & & & & & & \\ & & & C_2 & B_{21} & B_{23} & & & \\ & & & & & & C_3 & B_{31} & B_{32} \end{pmatrix}, \tag{6.3}$$

where c_i $(1 \leq i \leq 3)$ is a matrix defined similarly for the $\mathscr{H}_r$-matrix $E_{I_i \times I_i}$.

It is straightforward to extend the construction of C to a general d. If E is a 1×1 matrix, then if E is nonzero, we define $C = E$ and if E is zero, we set $C = [\,]$.

Now, suppose that $E_{I_1 \times I_1} = C_1 D_1$ is a k_1-sparse factorization. Then, we can write:

$$\begin{pmatrix} E_{I_1 \times I_1} \\ E_{I_2 \times I_1} \\ E_{I_3 \times I_1} \end{pmatrix} = \begin{pmatrix} C_1 & B_{12} & B_{13} & & & & & & \\ & & & C_2 & B_{21} & B_{23} & & & \\ & & & & & & C_3 & B_{31} & B_{32} \end{pmatrix} \begin{pmatrix} D_1 \\ 0 \\ 0 \\ 0 \\ D_{21} \\ 0 \\ 0 \\ D_{31} \\ 0 \end{pmatrix},$$

so that $E_{I_i \times I_1} = B_{i1} D_{i1}$.

Therefore, since in the matrix C, at the right of each c_i there are $d-1$ submatrices B_{ij} with at most r columns, we deduce from the above factorization that there is a k-sparse factorization $E = CD$ of E with $k = \max_{1 \leq i \leq d}\{k_i\} + rd^2$. Repeating this recursively over the h levels of $E_{I_i \times I_i}$ shows that $k = rd^2(h+1)$.

In Figure 6.2, we show an example of such a factorization for an $\mathscr{H}_1(\mathscr{P})$-matrix.

The two following properties on the matrix C will be of interest in the next sections.

Lemma 6.7. *The number of columns in C is given by $p = \mathcal{O}(rd^2 n)$.*

Proof. By induction on n, we prove that $p \leq rd(d-1)\left(\frac{d}{d-1}n - \frac{1}{d-1}\right)$.

- If $n = 2$, then $d = 2$ and $p \leq 4 \leq rd(d-1)\left(\frac{d}{d-1}n - \frac{1}{d-1}\right)$.

- If $n > 2$, then E is of the form (6.2). Below, the size of a diagonal block $E_{I_i \times I_i}$ is denoted by n_i. Note that $n = \sum_{i=1}^{d} n_i$.

  By construction of C, we know that $p \leq r(d-1)d + \sum_{i=1}^{d} p_i$, where p_i is the maximum number of columns in the matrix C_i of $E_{I_i \times I_i}$ $(1 \leq i \leq d)$. Consequently, by induction we have

$$p \leq r(d-1)d + r(d-1)d \sum_{i=1}^{d} \left(\frac{d}{d-1} n_i - \frac{1}{d-1} \right) = r(d-1)d \left(\frac{d}{d-1} n - \frac{1}{d-1} \right).$$

□

Lemma 6.8. *The cost of computing $C^T C$ is $\mathcal{O}(nr^2d^2h^2)$.*

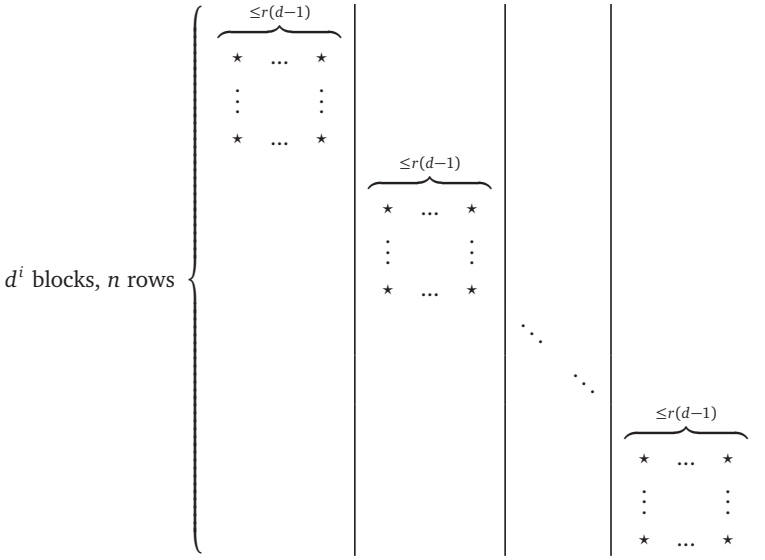
Proof. Since $C^T C$ is symmetric, it is sufficient to compute the upper part (including the diagonal) of $C^T C$. Consequently, for each column c_i of C, we have to compute all the scalar product $c_i^T c_j$ where $j \geq i$ and the support of c_j overlaps with the one of c_i.

From (6.3), consider the $d = 3$ following blocks

$$\begin{pmatrix} B_{12} & B_{13} \end{pmatrix}, \begin{pmatrix} B_{21} & B_{23} \end{pmatrix}, \begin{pmatrix} B_{31} & B_{32} \end{pmatrix}.$$

Each of these blocks has at most $r(d-1)$ columns and they are such that the support of a column of one block does not overlap with the support of a column in another block. More generally and recursively in the structure of C, we deduce that for each $1 \leq i \leq h+1$, C contains d^i blocks of at most $r(d-1)$ columns each such that the support of a column of one block does not overlap with the support of a column in another block. In other words, if we write these

d^i blocks consecutively, we get:



(6.4)

Note that these $\sum_{i=1}^{h+1} d^i$ blocks partition the columns of C, i.e. each column of C is contained in exactly one block. For the following reasoning, we consider the d^i blocks as shown in (6.4). For each column c_k of C contained in one of these d^i blocks, it follows from the recursive structure of C that we have to compute $\mathcal{O}(r(d-1)i)$ scalar products of the form $c_k^T c_j$ where c_j is a column of C with $j \geq k$. Due to the structure of (6.4), it follows that the total cost of the scalar products for all the columns contained in the d^i blocks is $\mathcal{O}(nr^2d^2i)$. Indeed, this is can be interpreted as computing in $\mathbb{R}^n$ $\mathcal{O}(r(d-1) \times r(d-1)i) = \mathcal{O}(r^2d^2i)$ scalar products, since each block has $\mathcal{O}(r(d-1))$ columns and we have to compute $\mathcal{O}(r(d-1)i)$ scalar products for each of them.

Consequently, the total cost for C^TC is given by

$$\mathcal{O}\left(\sum_{i=1}^{h+1} nr^2d^2i\right) = \mathcal{O}\left(nr^2d^2h^2\right).$$

□

Theorem 6.9. *Let $E \in \mathbb{R}^{n\times n}$ be an $\mathcal{H}_r(\mathcal{P})$-matrix with a dendrogram $\mathcal{P}$ of depth h and maximum degree d. Then, there are matrices $C \in \mathbb{R}^{n\times p}$ and $D \in \mathbb{R}^{p\times n}$ such that the factorization $E = CD$ is k-sparse for $k = rd^2(h+1)$. Moreover, $p = \mathcal{O}(rd^2n)$ and $Cost(C^TC) = \mathcal{O}(nr^2d^2h^2)$.*

Throughout the chapter, in a k-sparse factorization $E = CD$ of an $\mathscr{H}_r(\mathscr{P})$-matrix, the matrix C is supposed to be of the form (6.3) generalized to any d. Moreover, such a matrix C will be called *C-matrix*.

6.2.3 Examples of hierarchical matrices

Below, we give two examples of hierarchical matrices. The first one is about semiseparable matrices, whereas the second one is obtained from graph theory.

Example 1: semiseparable matrices

We prove that a (p,q)-semiseparable matrix (see Definition 6.10) is an $\mathscr{H}_r(\mathscr{P})$-matrix where $r = \max\{p,q\}$ and $\mathscr{P}$ is a binary dendrogram, i.e. $d = 2$.

We mention that this example is given for information purposes only. Indeed, semiseparable matrices are so structured that there already exist algorithms that solve in linear time systems involving them [98, 99]. Although the method presented in this chapter is therefore of interest for hierarchical matrices that are not semiseparable, we think that it is worth mentioning that semiseparable matrices are among the hierarchical ones.

Definition 6.10. *[98] An $n \times n$ matrix E is called $\{p,q\}$-semiseparable if the following relations are satisfied:*

$$rank(E(1 : i+q-1, i : n)) \leq q \text{ and } rank(E(i : n, 1 : i+p-1)) \leq p$$

for all feasible $1 \leq i \leq n$.

Theorem 6.11. *An $n \times n$ matrix that is $\{p,q\}$-semiseparable is an $\mathscr{H}_r(\mathscr{P})$-matrix where $r = \max\{p,q\}$ and $\mathscr{P}$ is a binary dendrogram, namely $d = 2$.*

Proof. Following the definition, we have $1 \leq p,q \leq n$ and we assume without loss of generality that $n \geq 2$.

Let E be an $n \times n$ matrix which is (p,q)-semiseparable and let $\{I_1, I_2\}$ be a partition of $\{1,...,n\}$ with $I_1 = \{1,...,\lfloor \frac{n}{2} \rfloor\}$ and $I_2 = \{1,...,n\} \backslash I_1$.

Consider an integer $i_1 \in \{1,...,n\}$ such that $\lfloor \frac{n}{2} \rfloor - q + 1 \leq i_1 \leq \min\{\lfloor \frac{n}{2} \rfloor, n - q + 1\}$. Then, the submatrix $E(1 : i_1 + q - 1, i_1 : n)$ which is of rank $\leq q$ contains $E_{I_1 \times I_2}$.

Similarly, if i_2 is an integer of $\{1,...,n\}$ such that $\lfloor \frac{n}{2} \rfloor - p + 1 \leq i_2 \leq \min\{\lfloor \frac{n}{2} \rfloor, n - p + 1\}$, then the submatrix $E(i_2 : n, 1 : i_2 + p - 1)$ which is of rank $\leq p$ contains $E_{I_2 \times I_1}$.

Therefore, we have shown that the off-diagonal blocks $E_{I_1\times I_2}$ and $E_{I_2\times I_1}$ are of rank $\leq r$.

From the definition of semiseparable matrix, it follows that the diagonal blocks $E_{I_1\times I_1}$ and $E_{I_2\times I_2}$ of E are also (p,q)-semiseparable matrices.

Repeating the previous argument recursively on $E_{I_1\times I_1}$ and $E_{I_2\times I_2}$ shows that E is an $\mathscr{H}_r(\mathscr{P})$-matrix with $\mathscr{P}$ being a binary dendrogram. □

Whereas a (p,p)-semiseparable matrix is an $\mathscr{H}_p$-matrix, the reverse does not hold. The $\mathscr{H}_1$-matrix shown in Figure 6.2 is a good example.

Example 2: reduced incidence matrices of trees and their inverse

In what follows, we prove that a reduced incidence matrix of a tree (see Definition 6.12) is permutation similar to an $\mathscr{H}_1(\mathscr{P})$-matrix where $\mathscr{P}$ is a binary dendrogram, meaning that $d = 2$.

Definition 6.12. *A reduced incidence matrix of a tree T is an incidence matrix of T from which one row has been removed.*

The two following lemmas are used in the proof of Proposition 6.15, stating that a reduced incidence matrix of a tree is permutation similar to an $\mathscr{H}_1$-matrix. The first lemma is known under the name *Tree Vertex Separator lemma* [53,56].

Lemma 6.13 (Tree Vertex Separator, [53, 56])**.** *For a tree T rooted at s with $n \geq 2$ nodes, in $\mathscr{O}(n)$ time one can compute a node d that divides T into edge-disjoint subtrees $T_0, ..., T_k$ in such a way that:*

- *each T_i has at most $\frac{n}{2} + 1$ nodes*
- *T_0 is rooted at s and contains d as a leaf*
- *$T_1, ..., T_k$ are rooted at d.*

Lemma 6.14. *Let $n \geq 2$ be a natural number and let a partition of $\{1, ..., n\}$ where each set has at most $\lceil \frac{n}{2} \rceil$ elements. Then, one can separate these sets into two groups with at most $\frac{2}{3}n$ elements per group.*

Proof. Let $S_1, ..., S_k$ be a partition of $\{1, ..., n\}$ with each S_i having at most $\lceil \frac{n}{2} \rceil$ elements. It is sufficient to prove that one can gather some S_i's in order to get a set where the number of elements is between $\frac{n}{3}$ and $\frac{2}{3}n$.
Claim: for each natural number $n \geq 2$, $\lceil \frac{n}{2} \rceil \leq \frac{2}{3}n$.
Proof of the claim: If n is even, the result is clear. If n is odd, then $\lceil \frac{n}{2} \rceil = \frac{n+1}{2}$ and a short computation shows that $\frac{n+1}{2} \leq \frac{2}{3}n$ if and only if $n \geq 3$. □

Therefore, the above claim states that each S_i has at most $\frac{2}{3}n$ elements.

If there is a set S_i for which the size is between $\frac{n}{3}$ and $\frac{2}{3}n$, the proof is over. Otherwise, since each S_i has at most $\frac{2}{3}n$ elements, each S_i must have a size less than $\frac{n}{3}$. In such a case, gather two sets, say S_1 and S_2, in a bigger set $\tilde{S}$.

Either the size of $\tilde{S}$ is between $\frac{n}{3}$ and $\frac{2}{3}n$ and the proof is over, or $\tilde{S}$ has less than $\frac{n}{3}$ elements. In this case, repeat the above argument on the partition $\tilde{S}, S_3, ..., S_k$.

Since the S_i's partition $\{1, ..., n\}$, after a number of iterations, one gets a set $S_1 \cup S_2 \cup ... \cup S_i$ for which the size is between $\frac{n}{3}$ and $\frac{2}{3}n$.

□

Two matrices A and B are said to be permutation similar if there are two permutation matrices P_1 and P_2 such that $A = P_1 B P_2$.

Proposition 6.15. *A reduced incidence matrix of a tree is permutation similar to an $\mathscr{H}_1(\mathscr{P})$-matrix where $\mathscr{P}$ is a binary dendrogram, namely $d = 2$, with depth $h = \mathcal{O}(\log n)$.*

Proof. Let $E \in \mathbb{R}^{n\times n}$ be a reduced incidence matrix of a tree T on $n+1$ nodes.

We number the nodes and edges of T so that a resulting incidence matrix of T is of the form:

$$\begin{pmatrix} v^T \\ E \end{pmatrix},$$

for some vector $v \in \mathbb{R}^n$. Choose node 1 to be the root of T. Let d be a node of T as in Lemma 6.13 (possibly node d is equal to node 1). Notice that node d is the unique common node of the subtrees $T_0, ..., T_k$. Moreover, since each of $T_0, ..., T_k$ has no more than $\frac{n+1}{2} + 1$ nodes, each $T_i \backslash \{d\}$ has at most $\frac{n+1}{2}$ nodes, or equivalently at most $\lceil \frac{n}{2} \rceil$ nodes. Therefore, following Lemma 6.14, one can separate them into two groups, say $\{T_0, ..., T_i\}$ and $\{T_{i+1}, ..., T_k\}$, so that the subtree G_1 (resp. G_2) spanned by the edges of $T_0, ..., T_i$ (resp. $T_{i+1}, ..., T_k$) has at most $\frac{2}{3}n + 1$ nodes. Notice that G_1 and G_2 share only node d and have no common edge.

Denote by n_1 and e_1 the number of nodes and edges in G_1 respectively. Number from 1 to n_1 the nodes of G_1 so that node 1 of T remains the first node in G_1, from $n_1 + 1$ to $n + 1$ the nodes of $G_2 \backslash \{d\}$, from 1 to e_1 the edges of G_1 and from $e_1 + 1$ to n the edges of G_2.

The incidence matrix of T resulting from the above numbering of the nodes

and edges is of the form:

$$\begin{pmatrix} v_1^T & v_2^T \\ \tilde{E}_1 & B \\ 0 & \tilde{E}_2 \end{pmatrix},$$

where $\tilde{E}_1$ is a reduced incidence matrix of subtree G_1, $\tilde{E}_2$ is a reduced incidence matrix of subtree G_2 and B contains at most one nonzero row (the one corresponding to node d). Therefore, B has a rank of at most 1. Moreover, E is permutation similar to

$$\tilde{E} = \begin{pmatrix} \tilde{E}_1 & B \\ 0 & \tilde{E}_2 \end{pmatrix},$$

where the size of $\tilde{E}_1$ and $\tilde{E}_2$ is at most $\frac{2}{3}n$.

We repeat this recursively on the reduced incidence matrices $\tilde{E}_1$ and $\tilde{E}_2$ and E has been shown to be permutation similar to an $\mathscr{H}_1(\mathscr{P})$-matrix where $\mathscr{P}$ is a binary dendrogram. Moreover, since the size of each diagonal block is divided by at least $\frac{3}{2}$ at each step, the depth h of $\mathscr{P}$ is $\mathcal{O}(\log n)$. □

Given a reduced incidence matrix E, we deduce from the above proof and the Tree Vertex Separator lemma that it takes $\mathcal{O}(n \log n)$ time for computing a matrix that is an $\mathscr{H}_1(\mathscr{P})$-matrix permutation similar to E, as well as the dendrogram $\mathscr{P}$.

By a simple induction on the number on nodes in the tree, we can prove that a reduced incidence matrix is invertible. In the next proposition, we show that the inverse of a reduced incidence matrix is also permutation similar to an $\mathscr{H}_1(\mathscr{P})$-matrix.

Proposition 6.16. *The inverse of a reduced incidence matrix of a tree is permutation similar to an $\mathscr{H}_1(\mathscr{P})$-matrix, where $\mathscr{P}$ is a binary dendrogram.*

Proof. From the proof of Proposition 6.15, we know that a reduced incidence matrix is permutation similar to an invertible $\mathscr{H}_1$-matrix E of the form:

$$E = \begin{pmatrix} E_{I_1 \times I_1} & E_{I_1 \times I_2} \\ & E_{I_2 \times I_2} \end{pmatrix},$$

where $E_{I_1 \times I_2}$ has at most one nonzero row and $E_{I_1 \times I_1}$ and $E_{I_2 \times I_2}$ are invertible $\mathscr{H}_1$-matrices with the same recursive structure (since they are also reduced incidence matrices).

Below, we prove that the inverse of such of matrix E is an $\mathscr{H}_1$-matrix as well.

A short computation shows that

$$E^{-1} = \begin{pmatrix} E^{-1}_{I_1 \times I_1} & F \\ & E^{-1}_{I_2 \times I_2} \end{pmatrix},$$

where $F = -E^{-1}_{I_1 \times I_1} E_{I_1 \times I_2} E^{-1}_{I_2 \times I_2}$. In addition, $\text{rank}(F) \leq \text{rank}(E_{I_1 \times I_2}) \leq 1$. Therefore, this recursively repeated on $E_{I_1 \times I_1}$ and $E_{I_2 \times I_2}$ shows that E^{-1} is an $\mathscr{H}_1$-matrix. □

6.3 A linear solver with fast iterations

Theorem 6.3 allows us to efficiently approximate the minimal norm solution of under-determined linear systems. Given a compatible linear system $Ax = b$, we are looking for an approximation of

$$x^* := \arg \min_{s.t. Ax=b} ||x||, \tag{6.5}$$

where $||x|| := \sqrt{x^T x}$[§]. More precisely, given a precision $\epsilon > 0$, we are looking for a vector x^*_ϵ such that $Ax^*_\epsilon = b$ and

$$||x^*_\epsilon - x^*|| \leq \epsilon \cdot ||x^*||.$$

Such a vector $x^\star_\epsilon$ is then called an *ϵ-approximation* of $x^\star$.

This problem can be readily solved as follows. Suppose we are given matrix Q in which the columns form a basis of the kernel of A. If x_0 is a feasible solution to $Ax = b$, we write (6.5) as

$$x^* := \arg \min_{s.t. \ x=x_0+Q\cdot y} ||x||.$$

Consequently, we compute $x^\star_\epsilon$ by iteratively taking a column q of Q and by setting:

$$x_{t+1} = x_t + \alpha^\star q \qquad \text{with} \qquad \alpha^\star = \arg\min_{\alpha \in \mathbb{R}} ||x_t + \alpha q||.$$

A short computation shows that $\alpha^* = -\frac{x_t^T q}{q^T q}$ and each iteration is of the form (6.1). Note that each iteration starts at the current vector x_t and looks in the direction of a given vector q for the vector of minimal norm. The method should

[§]In the general case where the scalar products are of the form $x^T Rq$, then we use accordingly the R-norm, namely $||x||_R := \sqrt{x^T Rx}$. Recall that R denotes a positive definite diagonal matrix.

then be interpreted as a coordinate descent in the basis given by the columns of Q.

Since x_0 is a feasible solution and all updates added to x_0 are in the kernel of A, it should be noticed that each iterate x_t is an exact solution of $Ax = b$. Therefore, the above iterative method converges to the optimal x^*.

In the following theorem, we use this approach when A is an $n \times m$ matrix (with $n < m$) of the form $A = \begin{pmatrix} E & F \end{pmatrix}$, where E is an invertible $n \times n$ matrix. A matrix Q where the columns are a basis of the kernel of A is:

$$Q = \begin{pmatrix} E^{-1}F \\ -I_{m-n} \end{pmatrix}, \tag{6.6}$$

where I_{m-n} is the identity matrix of dimension $m-n$. Indeed, the fact that E is invertible implies that the rank of A is n. Therefore, we compute that $\dim\ker(A) = m - rank(A) = m - n$. Moreover, it is clear that $AQ = 0$ and that the $m-n$ columns of Q are linearly independent.

If E^{-1} is k-sparsely factorizable and F is f-column sparse, then we know by the second property of sparse factorization that $E^{-1}F$ is (kf)-sparse factorizable. As a consequence, if $E^{-1} = CD$ is a k-sparse factorization of E^{-1}, then we can write

$$Q = \tilde{C}\tilde{D} = \begin{pmatrix} C & 0 \\ 0 & I_{m-n} \end{pmatrix} \begin{pmatrix} DF \\ -I_{m-n} \end{pmatrix} \tag{6.7}$$

which is a $(1+kf)$-sparse factorization of Q.

By using the above arguments when E^{-1} is a hierarchical matrix, we show the following result from Theorems 6.3 and 6.9.

Theorem 6.17. *Let $A = \begin{pmatrix} E & F \end{pmatrix}$ be an $n \times m$ matrix with $n < m$, where $E \in \mathbb{R}^{n\times n}$ is invertible and E^{-1} is an $\mathscr{H}_r(\mathscr{P})$-matrix with an associated dendrogram $\mathscr{P}$ of maximum degree d and depth h. Further, let $F \in \mathbb{R}^{n\times(m-n)}$ be f-column sparse.*

Then, we can compute an ϵ-approximation of $x^* := \arg\min_{s.t\, Ax=b} ||x||$ by applying N_ϵ iterations of the form (6.1), in time

$$\mathscr{O}(N_\epsilon \cdot f\, r d^2 h + m f^2 r^2 d^4 h^2 + Cost(CD)),$$

where $Cost(CD)$ is the cost of computing a $(rd^2(h+1))$-sparse factorization of E^{-1}.

Proof. According to Theorem 6.9, let $E^{-1} = CD$ be a k-sparse factorization with $k = rd^2(h+1)$. By the first property on sparse factorization, we know that C

is k-row sparse and that each column of D is k-sparse. A feasible solution to $Ex = b$ is then computed in $\mathcal{O}(kn)$ time. Indeed, such a solution is given by $x_0 = E^{-1}b = CDb$.

Consider the matrix Q given in (6.6). By the observations made before stating the theorem, we know that the columns of Q are a basis of the kernel of A and that the matrix Q is $(1+kf)$-sparsely factorizable. Let $Q = \tilde{C}\tilde{D}$ be the $(1+kf)$-sparse factorization given in (6.7).

The cost of computing $\tilde{C}^T\tilde{C}$ is equal to the cost of computing C^TC. Indeed, we compute that

$$\tilde{C}^T\tilde{C} = \begin{pmatrix} C^T & 0 \\ 0 & I_{m-n} \end{pmatrix}\begin{pmatrix} C & 0 \\ 0 & I_{m-n} \end{pmatrix} = \begin{pmatrix} C^TC & 0 \\ 0 & I_{m-n} \end{pmatrix}.$$

As explained before stating the theorem, we start from the vector $\begin{pmatrix} x_0 \\ 0 \end{pmatrix}$ which is an exact solution to $Ax = b$ and iteratively we pick a column q of Q and we perform an iteration of the form (6.1). Theorem 6.3 applied with $Q, \tilde{C}$ and $\tilde{D}$ shows that the running time is given by

$$\mathcal{O}(N_\epsilon \cdot kf + pkf + mk^2f^2 + Cost(C^TC) + Cost(CD)).$$

Finally, we conclude the proof by using that $k = rd^2(h+1)$, $Cost(C^TC) = \mathcal{O}(nr^2d^2h^2)$ and $p = \mathcal{O}(rd^2n)$ (see Theorem 6.9). □

The number N_ϵ of needed iterations depends on the matrix A and on the way a column of Q is selected at each iteration. Estimating N_ϵ is an issue that we do not consider in this chapter.

In the following lemma, we show that approximating the minimal norm solution to a system $Bx = b$ where B is an incidence matrix of a connected and simple undirected graph G can be reduced to computing the minimal norm solution of a system like in the previous theorem. More specifically, the matrix E will be the reduced incidence matrix of a spanning tree of G. We recall that the inverse of a reduce incidence matrix is an $\mathscr{H}_1$-matrix (see Proposition 6.16).

Lemma 6.18. *Let B be an incidence matrix of a connected and simple undirected graph G on n nodes and let a system $Bx = b$. Then, given a spanning tree of G and a reduced incidence matrix E of this tree, there is a system of the form $\begin{pmatrix} E & F \end{pmatrix}x = b'$ that has the same solution set as $Bx = b$.*

Proof. Let T be a spanning tree of G. The incidence matrix B of G is then written

as:

$$B = \begin{pmatrix} v_{tree}^T & v^T \\ E_{tree} & F \end{pmatrix},$$

where $B_{tree} = \begin{pmatrix} r_{tree}^T \\ E_{tree} \end{pmatrix}$ is an incidence matrix of T and thus E_{tree} is a reduced incidence matrix of T.

A simple induction on the number n of nodes in a tree T shows that an incidence matrix B_{tree} and a reduced incidence matrix E_{tree} of T are of rank $n-1$.

Therefore, we deduce that $\text{rank}(B) = \text{rank}(B_{tree}) = n-1$ and that the submatrix $\begin{pmatrix} E_{tree} & F \end{pmatrix}$ has a rank equal to $n-1$.

The row $\begin{pmatrix} v_{tree}^T & v^T \end{pmatrix}$ is then a linear combination of the rows of $\begin{pmatrix} E_{tree} & F \end{pmatrix}$ and consequently, a vector x is a solution to $Bx = b$ if and only if x is a solution to $\begin{pmatrix} E_{tree} & F \end{pmatrix} x = b'$, where $b = \begin{pmatrix} \star \\ b' \end{pmatrix}$. □

Given a connected graph G, we can compute by using Kruskal's algorithm [63] a spanning tree of G in $\mathcal{O}(m \log n)$ time and we know that given a reduced incidence matrix E of the tree, it takes $\mathcal{O}(n \log n)$ time to compute a dendrogram $\mathscr{P}$ with maximum degree $d = 2$ and depth $h = \mathcal{O}(\log n)$, and an $\mathscr{H}_1(\mathscr{P})$-matrix $\tilde{E}$ that is permutation similar to E. In the framework of Lemma 6.18, we can then consider that E is such an $\mathscr{H}_1(\mathscr{P})$-matrix and that F is 2-column sparse. Therefore, from Theorem 6.17, we deduce that we can approximate the minimal norm solution to a compatible system $Bx = b$ in time

$$\mathcal{O}(N_\epsilon \cdot \log n + m \log^2 n + Cost(CD)),$$

where m is the number of edges in G. Below, we prove that we can compute in $\mathcal{O}(n \log^2 n)$ time a sparse factorization of the inverse of the $\mathscr{H}_1(\mathscr{P})$-matrix permutation similar to E.

Proposition 6.19. *Let E be an $n \times n$ $\mathscr{H}_1(\mathscr{P})$-matrix permutation similar to a reduced incidence matrix of a tree. Then, a representation of E^{-1} is computable in $\mathcal{O}(n \log^2 n)$ time.*

Proof. We know that E is of the form:

$$E = \begin{pmatrix} E_{I_1 \times I_1} & E_{I_1 \times I_2} \\ & E_{I_2 \times I_2} \end{pmatrix},$$

where an l^{th} row of $E_{I_1 \times I_2}$ may be nonzero and each diagonal block $E_{I_i \times I_i}$ is an

$\mathscr{H}_1$-matrix permutation similar to a reduced incidence matrix of a tree. Moreover, we know that the dendrogram $\mathscr{P}$ has depth $h = \mathscr{O}(\log n)$ (see Proposition 6.15).

The inverse of E is of the form:

$$E^{-1} = \begin{pmatrix} E^{-1}_{I_1\times I_1} & F \\ & E^{-1}_{I_2\times I_2} \end{pmatrix},$$

where $F = -E^{-1}_{I_1\times I_1} E_{I_1\times I_2} E^{-1}_{I_2\times I_2}$.

It follows that each column of F is a multiple of the l^{th} column of $E^{-1}_{I_1\times I_1}$, i.e.

$$F = (E^{-1}_{I_1\times I_1})_{\star l} \cdot \begin{pmatrix} \beta_1 & \dots & \beta_t \end{pmatrix},$$

where $(E^{-1}_{I_1\times I_1})_{\star l}$ denotes the l^{th} column of $E^{-1}_{I_1\times I_1}$. A representation of E^{-1} keeps in memory the vector $\beta := \begin{pmatrix} \beta_1 & \dots & \beta_t \end{pmatrix}$ and the index l. It follows from $F = -E^{-1}_{I_1\times I_1} E_{I_1\times I_2} E^{-1}_{I_2\times I_2}$ that this vector β is computed by solving the system:

$$-\begin{pmatrix} \alpha_1 & \dots & \alpha_t \end{pmatrix} \cdot E^{-1}_{I_2\times I_2} = \begin{pmatrix} \beta_1 & \dots & \beta_t \end{pmatrix},$$

where $\begin{pmatrix} \alpha_1 & \dots & \alpha_t \end{pmatrix}$ is the l^{th} row of $E_{I_1\times I_2}$. Or equivalently, vector β is obtained by solving the triangular system:

$$-E^T_{I_2\times I_2} \cdot \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_t \end{pmatrix} = \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_t \end{pmatrix}.$$

Since the elementary blocks of E have at most one nonzero row, we deduce that each row of $E^T_{I_2\times I_2}$ is h-sparse. Consequently, it takes $\mathscr{O}(nh)$ time to solve it.

Similarly, in $E^{-1}_{I_1\times I_1}$ and $E^{-1}_{I_2\times I_2}$, we compute a representation of F_1 and F_2, defined respectively in $E^{-1}_{I_1\times I_1}$ and $E^{-1}_{I_2\times I_2}$ as F is defined in E^{-1}, in time $\mathscr{O}(t_1 h)$ and $\mathscr{O}(t_2 h)$ respectively, with $t_1 + t_2 \leq n$. Therefore, it takes in total for computing a representation of F_1 and F_2 $\mathscr{O}(nh)$ time.

Since the number of iterations is at most $h+1$, it takes $\mathscr{O}(nh^2)$ time for computing such a representation of E^{-1}. Since $h = \mathscr{O}(\log n)$, the proof is over. □

Proposition 6.20. *Let E be an $n \times n$ $\mathscr{H}_1(\mathscr{P})$-matrix permutation similar to a reduced incidence matrix of a tree. Given the representation of E^{-1} provided by the proof of Proposition 6.19, a C-matrix of E^{-1} is computable in $\mathscr{O}(n \log^2 n)$ time.*

Proof. Since E is upper triangular, the diagonal of E^{-1} is computed in $\mathcal{O}(n)$ time.

We denote by h the height of $\mathscr{P}$, which we know to be equal to $\mathcal{O}(\log n)$. We recall also that the maximum degree of $\mathscr{P}$ is $d = 2$.

Given the representation and the diagonal of E^{-1}, each entry in the upper part of E^{-1} can be computed in $\mathcal{O}(h)$ time. Indeed, a such entry is of the form $\beta_{i_1}...\beta_{i_t} e$ where e is a diagonal entry of $E^{-1}$and the $\beta_{i_1}, ..., \beta_{i_t}$ are $\mathcal{O}(h)$ coefficients computed for the representation of E^{-1}.

Recall from the proof of Proposition 6.19 that

$$E^{-1} = \begin{pmatrix} E^{-1}_{I_1 \times I_1} & F \\ & E^{-1}_{I_2 \times I_2} \end{pmatrix}$$

where every column of F is a multiple of an l^{th} column of $E^{-1}_{I_1 \times I_1}$. Then, during the construction of a C-matrix $C \in \mathbb{R}^{n \times p}$ of E^{-1}, we choose as a basis of the column space of F, if it is not reduced to $\{0\}$, the l^{th} column of $E^{-1}_{I_1 \times I_1}$.

Therefore, each nonzero entry of C is an entry of E^{-1}. Moreover, since E is $\mathcal{O}(\log n)$-sparse factorizable (see Theorem 6.9), by the first property of sparse factorization we know that C is $\mathcal{O}(\log n)$-row sparse. Consequently, C has $\mathcal{O}(nh)$ nonzero entries and we compute them in $\mathcal{O}(nh^2)$ time. Since $h = \mathcal{O}(\log n)$, the proof is over. □

Proposition 6.21. *Let E be an $n \times n$ $\mathscr{H}_1(\mathscr{P})$-matrix permutation similar to a reduced incidence matrix of a tree. Given the representation of E^{-1} provided by the proof of Proposition 6.19 and the C-matrix $C \in \mathbb{R}^{n \times p}$ of E^{-1} provided by the proof of Proposition 6.20, we compute in $\mathcal{O}(n \log n)$ time a matrix $D \in \mathbb{R}^{p \times n}$ such that $E^{-1} = CD$ is a $\mathcal{O}(\log n)$-sparse factorization of E^{-1}.*

Proof. This follows from the representation of E^{-1}, the C-matrix C and the structure of D as presented in Section 3.2. □

Corollary 6.22. *Given an $n \times n$ $\mathscr{H}_1(\mathscr{P})$-matrix E permutation similar to a reduced incidence matrix of a tree, a sparse decomposition of E^{-1} is computable in $\mathcal{O}(n \log^2 n)$ time.*

We have thus proved the following theorem.

Theorem 6.23. *Let $B \in \mathbb{R}^{n \times m}$ be an incidence matrix of a connected and simple undirected graph G on n nodes and m edges. Then, we can compute an ϵ-approximation of $x^* := \arg\min_{s.t. Bx = b} ||x||$ by N_ϵ iterations of the form (6.1) in*

time

$$\mathcal{O}(N_\epsilon \cdot \log n + m \log^2 n).$$

Using the algorithm of [2], we can compute in $\mathcal{O}(m \log n)$ time a so-called low-stretch spanning tree. Moreover, following [56], if we define τ as $\tau = \text{trace}\left(Q^T Q\right)$ (depending on the stretch of the tree) and the probability to pick a column q of Q by

$$pr(q) := \frac{q^T q}{\tau},$$

then the number of needed iterations to get a vector $x^\star_\epsilon$ such that

$$\mathbb{E}\left[||x^\star_\epsilon - x^\star||\right] \leq \epsilon \cdot ||x^\star||$$

is $N_\epsilon = \mathcal{O}(m \log n \log(n/\sqrt{\epsilon}))$.

Consequently, our algorithm offers another way to obtain the following result, already appeared in [56].

Corollary 6.24. *[56, 95] Let $Bx = b$ be a compatible system where $B \in \mathbb{R}^{n \times m}$ is an incidence matrix of a connected graph. Then, given a precision $\epsilon > 0$, there is a probabilistic method providing in time $\mathcal{O}(m \log^2 n \log(n/\sqrt{\epsilon}))$ a vector $x^\star_\epsilon$ such that*

$$\mathbb{E}\left[||x^\star_\epsilon - x^\star||\right] \leq \epsilon \cdot ||x^\star||,$$

where $x^\star$ is the minimal norm solution to $Bx = b$.

6.4 Application: Laplacian systems

The goal of this section is to apply Corollary 6.24 in order to solve a Laplacian system in nearly-linear time. By Laplacian system, we mean a compatible system $Ly = b$ where L is the Laplacian matrix of a simple undirected graph.

Given a Laplacian system, we would like to solve it in $\mathcal{O}(m \log^c n)$ time, where n and m respectively denote the number of nodes and edges in the graph and c is an arbitrary constant. Specifically, given $\epsilon > 0$, we are looking for a vector $y^\star_\epsilon$ such that

$$||y^\star_\epsilon - y^\star||^2_L \leq \epsilon \cdot ||y^\star||^2_L, \tag{6.8}$$

where $Ly^\star = b$ and $||y||^2_L := y^T L y$.

To do so, we factorize the Laplacian matrix L as $L = BB^T$, where B is an incidence matrix of the graph. Then, we set $x = B^T y$ and by using our al-

gorithm with a low-stretch spanning tree T and a precision $\tilde{\epsilon} := \epsilon/\tau$ where $\tau = \text{trace}\left(Q^T Q\right)$, we approximate the minimal norm solution to $Bx = b$. After a number $N_{\tilde{\epsilon}} = \mathcal{O}(m \log n \log(n/\sqrt{\tilde{\epsilon}}))$ of iterations, we get a vector $x^\star_{\tilde{\epsilon}}$ such that

$$\mathbb{E}\left[||x^\star_{\tilde{\epsilon}} - x^\star||\right] \leq \tilde{\epsilon} \cdot ||x^\star||.$$

Then, we compute the minimal Euclidean[¶] norm solution $y^\star_\epsilon$ to

$$x^\star_{\tilde{\epsilon}}(T) = B^T_{tree} y,$$

where B_{tree} denotes the incidence matrix, contained in B, of the spanning tree T and $x^\star_{\tilde{\epsilon}}(T)$ denotes the restriction of $x^\star_{\tilde{\epsilon}}$ to the edges of T.

It should be noticed that a system of the form $x = B^T_{tree} y$ can be solved in linear time. Indeed, start from an arbitrary node of the tree and give to it an arbitrary value, then go through the tree by a breath-first-search and determine the value of each node successively. From this and the fact that $\dim \ker(B^T_{tree}) = 1$, we see that $y^\star_\epsilon$ can be computed in $\mathcal{O}(n)$ time. This second part is different from the one of [56], however we prove with similar arguments that the resulting vector $y^\star_\epsilon$ satisfies

$$\mathbb{E}\left[||y^\star_\epsilon - y_\epsilon||^2_L\right] \leq \epsilon \cdot ||y^\star||^2_L, \tag{6.9}$$

for each vector $y^\star$ such that $x^\star = B^T y^\star$ (since $(\ker B)^\perp = Im(B^T)$, such vectors $y^\star$ exist).

Finally, we note that we obtained a vector $y^\star_\epsilon$ in $\mathcal{O}(m \log^2 n \log(n/\sqrt{\tilde{\epsilon}}))$ time.

6.5 Conclusion

In this chapter, we have contributed to the problem of solving a linear system in nearly-linear time. Indeed, we have presented a k-sparse matrix factorization which allows to compute in $\mathcal{O}(k)$ time each iteration of the form

$$x_{t+1} = x_t - \frac{x_t^T q}{q^T q} \cdot q,$$

where q is a column of a k-sparsely factorizable matrix.

[¶]It is straightforward to generalize the method to a graph with positive weights along the edges. In that case, the Laplacian matrix is factorized as $L = BRB^T$ where R is the diagonal matrix of the weights. The norm $||x||$ on $\mathbb{R}^m$ is then replaced by the R-norm $||x||_R$. However, in this second step of the method, we keep the Euclidean norm.

Moreover, we have shown that the hierarchical matrices are k-sparsely factorizable with k depending on the depth and the height of the hierarchical structure, but not directly on the size of the matrix.

From this, we have deduced an iterative method with fast iterations that approximates the minimal norm solution of under-determined linear systems. In particular, we have shown that we can use this approach when the coefficient matrix is the incidence matrix of a connected and simple undirected graph.

Finally, we have deduced a method to solve Laplacian systems in nearly-linear time.

Consequently, this chapter provides a new approach for nearly-linear time solvers and paves the way to further progress in that problem. Here are some open questions:

- What can be said about the number N_ϵ of needed iterations in Theorem 6.17 ?
- What are other examples of $\mathscr{H}_r$-matrices ?
- Can we identify matrices with an inverse that is an $\mathscr{H}_r$-matrix ?
- What would be other applications of Theorem 6.17 ?

Chapter 7

Conclusions

In this thesis, we have explored four different topics of combinatorial matrix theory, ranging from minimum rank problems and structural controllability to binary factorizations and fast linear solvers. For each of these topics, we look back and summarize our contributions before concluding by a personal view on the subject.

Our first contributions were devoted to minimum rank problems and especially to the computation of the zero forcing number, a graph invariant recently defined in order to compute the minimum rank of each kind of tree.

Our first result extends the one of [1] that computing the zero forcing number of a simple (directed or undirected) graph is NP-hard and proves that the computation of this invariant in a loop directed graph is also NP-hard.

In the literature, it is said that the zero forcing number solves the minimum rank problem of each type of tree. However, although it is clear how the zero forcing number provides the minimum rank of the tree, whatever its type, there exist few efficient algorithms computing the zero forcing number of a given type of tree: the only known algorithms are for simple or loop (undirected) trees. Our second result fills some of this gap and identifies a class of loop directed trees for which the zero forcing number can be computed in linear time using a simple elimination process.

As a consequence, our contributions of Chapter 3 first emphasize the NP-hardness of computing the zero forcing number in loop directed graphs and then, emphasize the lack of efficient algorithms for the zero forcing number of loop or simple directed trees.

While working on the computation of the zero forcing number, we tried to find an efficient way to compute the zero forcing number of every loop directed tree. Initially, we tried to work on the tree itself by following a suggestion given in [47]. However, since working directly on the tree was not conclusive, we had the idea to work on a hypergraph that gathers in a hyperedge the out-neighbors of a node i by forgetting the source node i. Specifically, the nodes of the hyper-

graph were the nodes of the tree and the out-neighbors of each node in the tree were considered as an hyperedge. Consequently, the hypergraph contains less information than the tree and our hope was to easily identify the zero forcing number in this poorer structure. However, even with this representation, we were not able to find an efficient algorithm. Today, we even wonder if such algorithms do exist. As a matter of fact, we also tried to figure out if computing the zero forcing number of a loop directed tree is NP-hard, but unfortunately our investigations did not bring any answer. As we have shown in Chapter 4, the zero forcing number also applies in other contexts than minimum ranks. Consequently, an answer to that question would be of important interest.

Our next contributions were thus devoted to the role of zero forcing in strong controllability. Throughout Chapter 4, we have considered networked systems underlying a loop directed graph and made the relation between the strong controllability of such systems and the zero forcing sets.

We have first provided a necessary and sufficient condition in terms of zero forcing for the strong controllability of a networked system from a given input set. Then, in the case of self-damped systems, we have proved that the minimum zero forcing sets in the simple interconnection graph are minimum input sets for strong controllability. Finally, we have deduced a polynomial time algorithm finding a minimum-size input set for the strong controllability of a self-damped system with a tree structure. The algorithm is one of those computing a minimum zero forcing set in a simple tree.

In the context of structural controllability, zero forcing sets are more than equivalent to constraint matchings. Indeed, they also give a natural and intuitive algorithm that checks if the system is strongly controllable from a given input set. As a consequence, zero forcing is according to us an appropriate notion to study dynamical systems. Furthermore, our feeling is confirmed by the results of [15, 77]. That is why we think that further research in this direction should be carried out.

A part of our contributions was devoted to the binary factorizations of the all ones matrix. More precisely, we were interested in the binary solutions to $A^m = J$, where J denotes the matrix of all ones. Although such a solution is known to share several characteristics with the De Bruijn matrices, which are the only known solutions to $A^m = J$, the relation between the De Bruijn matrices and a general binary solution to $A^m = J$ is far from being understood. The goal of Chapter 5 was to initiate research in this direction.

As preliminary results on this topic, we have first proved that each binary solution to $A^m = J$ with minimum rank is isomorphic to a row permutation of a

De Bruijn matrix. From this, we have partially solved Hoffman's open problem by characterizing, through the De Bruijn matrices, the minimum rank solutions to $A^2 = J$. Finally, we have identified a class of roots that are isomorphic to a De Bruijn matrix.

As explained in the introduction of Chapter 5, the binary solutions to $A^m = J$ are optimal strategies for the finite-time average consensus. Nowadays, the only optimal strategy we know is the De Bruijn matrix. Therefore, we think that other optimal strategies would be meaningful and hope that our results will motivate further research on the understanding of the binary roots of the all ones matrix.

Finally, our last contributions were about fast linear solvers. Specifically, we have given a new k-sparse matrix factorization that allows to compute in $\mathcal{O}(k)$ time each iteration of the form

$$x_{t+1} = x_t - \frac{x_t^T q}{q^T q} \cdot q,$$

provided q is a column of a k-sparsely factorized matrix. Moreover, we have applied this result to the resolution of a linear system. In particular, we have considered under-determined systems and shown that we can find the minimal norm solution with an iterative algorithm that has fast iterations. In addition, we have shown that our algorithm applies particularly well to systems where the coefficient matrix is an incidence matrix of a simple undirected graph. Indeed, the running time becomes then nearly-linear in the system size. Finally, we have deduced a nearly-linear time algorithm for Laplacian systems.

Compared to the current literature on nearly-linear time solvers, our sparse matrix factorization brings in our view a new perspective to the problem. Consequently, we hope that our contributions presented in Chapter 6 will pave the way to further progress in that research area. For instance, we think that our sparse matrix factorization might also be used in order to solve over-determined linear systems in nearly-linear time.

Bibliography

[1] A. Aazami. *Hardness results and approximation algorithms for some problems on graphs*. PhD thesis, University of Waterloo, 2008.

[2] I. Abraham and O. Neiman. Using petal-decompositions to build a low stretch spanning tree. *Proceedings of the* 44^{th} *Symposium on Theory Of Computing, STOC'12, New York, NY, USA*, pages 395–406, 2012.

[3] AIM minimum rank special work group. Zero forcing sets and the minimum rank of graphs. *Linear Algebra and its Applications*, 428:1628–1648, 2008.

[4] S. Ambikasaran and E. Darve. An $\mathcal{O}(N \log N)$ fast direct solver for partial hierarchically semi-separable matrices. *Journal of Scientific Computing*, 57(3):477–501, 2013.

[5] F. Barioli, S. Fallat, D. Hershkowitz, H.T. Hall, L. Hogben, H. van der Holst, and B. Shader. On the minimum rank of not necessarily symmetric matrices: a preliminary study. *Electronic Journal of Linear Algebra*, 18:126–145, 2009.

[6] F. Barioli, S. Fallat, and L. Hogben. Computation of minimal rank and path cover number for certain graphs. *Linear Algebra and its Applications*, 392:289–303, 2004.

[7] F. Barioli and S.M. Fallat. On the eigenvalues of generalized stars and double generalized stars. *Linear Algebra and its Applications*, 53:269–291, 2005.

[8] F. Barioli, S.M. Fallat, and L. Hogben. On the difference between the maximum multiplicity and path cover number for tree-like graphs. *Linear Algebra and its Applications*, 409:13–31, 2005.

[9] W. Barrett, H. van der Holst, and R. Loewy. Graphs whose minimal rank is two. *Electronic Journal of Linear Algebra*, 11:258–280, 2004.

[10] C. Berge. Two theorems in graph theory. *Proc. Nat. Acad. Sci. USA*, 43:842–844, 1957.

[11] A. Berman, S. Friedland, L. Hogben, U.G. Rothblum, and B. Shader. An upper bound for the minimum rank of a graph. *Linear Algebra and its Applications*, 429(7):1629–1638, 2008.

[12] R.A. Brualdi. *Combinatorial matrix classes*, volume 13. Cambridge University Press, 2006.

[13] R.A. Brualdi, L. Hogben, and B. Shader. AIM workshops on spectra of families of matrices described by graphs, digraphs and sign patterns, final report: Mathematical results, 2007. http://aimath.org/postworkshops/matrixspectrumrep.pdf.

[14] R.A. Brualdi and H.J. Ryser. *Combinatorial Matrix Theory*. Cambridge University Press, Cambridge, 1991.

[15] D. Burgarth, D. D'Alessandro, L. Hogben, S. Severini, and M. Young. Zero forcing, linear and quantum controllability for systems evolving on networks. *IEEE Transactions on Automatic Control*, 58:2349, 2013.

[16] M.K. Camlibel, S. Zhang, and M. Cao. Comments on "controllability analysis of multi-agent systems using relaxed equitable partitions". *International Journal of Systems, Control and Communications*, 4(112):72–75, 2012.

[17] A. Cayley. On the theory of the analytical forms called trees. *Philosiphical Magazine, Series IV*, 13(85):172–176, 1857.

[18] A. Cayley. Ueber die Analytischen Figuren, welche in der Mathematik Baüme genannt werden und ihre Anwendung auf die Theorie chemischer Verbindungen. *Berichte der deutschen Chemischen Gesellschaft*, 8(2):1056–1059, 1875.

[19] A. Chapman and M. Mesbahi. Strong structural controllability of networked dynamics. *Proc. American Control Conference*, pages 6141–6146, 2013.

[20] M.B. Cohen, R. Kyng, J.W. Pachocki, R. Peng, and A. Rao. Preconditioning in expectation. *CoRR abs/1401.6236*, 2014.

[21] C. Commault, J.M. Dion, and J.W. van der Woude. Characterization of generic properties of linear structured systems for efficient computations. *Kybernetika*, 38:503–520, 2002.

[22] N.J. Cowan, E.J. Chastain, D.A. Vilhena, J.S. Freudenberg, and C.T. Bergstrom. Nodal dynamics, not degree distributions, determine the structural controllability of complex networks. *PLoS ONE*, 7(6):e38398, 2012.

[23] D. Cvetcović, P. Rowlinson, and S. Simić. *Eigenspaces of graphs*. Cambridge University Press, Cambridge, 1997.

[24] W. Dahmen and R. Schneider. Wavelets on manifolds I: construction and domain decomposition. *SIAM Journal on Mathematical Analysis*, 31(1):184–230, 1999.

[25] T.A. Davis. Direct methods for sparse linear systems. *SIAM*, 2006.

[26] N.G. de Bruijn. A combinatorial problem. *Proceedings of the Koninklijke Nederlandse Akademie van Wetenschappen*, 49:758–764, 1946.

[27] L.M. DeAlba, T.L. Hardy, I.R. Hentzel, L. Hogben, and A. Wangness. Minimum rank and maximum eigenvalue multiplicity of symmetric tree sign patterns. *Linear Algebra and its Applications*, 418:389–415, 2006.

[28] L. DeLoss, J. Grout, T. McKay, J. Smith, and G. Tims. Program for calculating bounds on the minimum rank of a graph using *Sage*. Available at http://arxiv.org/abs/0812.1616.

[29] J.-C. Delvenne, R. Carli, and S. Zampieri. Optimal strategies in the average consensus problem. *Systems and Control Letters*, 58:759–765, 2009.

[30] J. Edmonds. Paths, trees and flowers. *Canadian J. Math.*, 17:449–467, 1965.

[31] M. Egerstedt, S. Martini, M. Coa, M.K. Camlibel, and A Bicchi. Interacting with networks: How does structure relate to controllability in single-leader, consensus network ? *Control Systems Magazine*, 32(4):66–73, 2012.

[32] H.C. Elman. Iterative methods for linear systems. *Large-scale matrix problems and the numerical solution of partial differential equations*, 3:69–177, 1994.

[33] L. Euler. Solutio problematis ad geometriam situs pertinentis. *Commentarii academiae scientiarum Petropolitanae*, 8:128–140, 1736.

[34] S. Even and R.E. Tarjan. Network flow and testing graph connectivity. *SIAM journal on computing*, 4(4):507–518, 1975.

[35] S. Fallat and L. Hogben. The minimum rank of symmetric matrices described by a graph : A survey. *Linear Algebra and its Applications*, 426:558–582, 2007.

[36] M. Fiedler. A characterization of tridiagonal matrices. *Linear Algebra and its Applications*, 2:191–197, 1969.

[37] L.R. Ford and D.R. Fulkerson. *Flows in networks*. Princeton University Press, Princeton, N.J., 1962.

[38] J.H. Fowler and N.A. Christakis. Cooperative behavior cascades in human social networks. *Proceedings of the National Academy of Sciences*, 107(12):5334–5338, 2010.

[39] C.D. Godsil and S. Severini. Control by quantum dynamics on graphs. *Physical Review A*, 81(5):052316, 2010.

[40] M.C. Golumbic, T. Hirst, and M. Lewenstein. Uniquely restricted matchings. *Algorithmica*, 31(2):139–154, 2001.

[41] L. Grasedyc and W. Hackbusch. Construction and arithmetics of H-matrices. *Computing*, 70(4):295–334, 2003.

[42] W. Hackbusch. A sparse matrix arithmetic based on $\mathscr{H}$-matrices. part 1: introduction to $\mathscr{H}$-matrices. *Computing*, 62(2):89–108, 1999.

[43] H. Heesch. Untersuchungen zum Vierfarbenproblem. Mannheim: Bibliographisches Institut, 1979.

[44] D. Hershkowitz and H. Schneider. Ranks of zero patterns and sign patterns. *Linear Multilinear Algebra*, 34:3–19, 1993.

[45] A.J. Hoffman. Research problems 2-11. *J. Combin. Theory*, 2:393, 1967.

[46] L. Hogben. Spectral graph theory and the inverse eigenvalue problem of a graph. *Chamchuri Journal of Mathematics*, 1:51–72, 2009.

[47] L. Hogben. Minimum rank problems. *Linear Algebra and its Applications*, 432(8):1961–1974, 2010.

[48] John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal of Computing*, 2(4):225–231, 1973.

[49] C.R. Johnson and A. Leal-Duarte. The maximum multiplicity of an eigenvalue in a matrix whose graph is a tree. *Linear Multilinear Algebra*, 46:139–144, 1999.

[50] C.R. Johnson and A. Leal-Duarte. On the possible multiplicities of the eigenvalues of a hermitian matrix whose graph is a tree. *Linear Algebra and its Applications*, 348:7–21, 2002.

[51] C.R. Johnson, A. Leal-Duarte, and C.M. Saiago. Inverse eigenvalue problems and lists of multiplicities for matricies whose graph is a tree: the case of generalized stars and double generalized stars. *Linear Algebra and its Applications*, 373:311–330, 2003.

[52] C.R. Johnson and C.M. Saiago. Estimation of the maximum multiplicity of an eigenvalue in terms of the vertex degrees of the graph of the matrix. *Electronic Journal of Linear Algebra*, 9:27–31, 2002.

[53] C. Jordan. Sur les assemblages de lignes. *J. Reine Angew Math*, 70:185–190, 1869.

[54] R.E. Kalman. Contributions to the theory of optimal control. *Bol. Soc. Mat. Mexicana*, 5(2):102–119, 1960.

[55] R.M. Karp. Reducibility among combinatorial problems. *R.E. Miller and J.W. Thatcher, eds., Complexity of computer computations (Plenum Press, New York)*, pages 85–104, 1972.

[56] J.A. Kelner, L. Orecchia, A. Sidford, and Z. Allen Zhu. A simple, combinatorial algorithm for solving SDD systems in nearly-linear time. *Proceedings of the 45th annual ACM symposium on Symposium on theory of computing, STOC'13, New York, NY, USA*, pages 911–920, 2013.

[57] M.C. Kempton. *The minimum rank, inverse inertia, and inverse eigenvalue problems for graphs*. PhD thesis, Brigham Young University, August 2010.

[58] I.-J. Kim. Algorithmic proof of maxmult$(t) = p(t)$. *Communications of the Korean Mathematical Society*, 27(4):665–668, 2012.

[59] F King and K. Wang. On the g-circulant solutions to the matrix equation $A^m = \lambda J$. *Journal of Combinatorial Theory Ser. A*, 38(182-186), 1985.

[60] I. Koutis, G.L. Miller, and R. Peng. Approaching optimality for solving sdd linear systems. *51st Annual Symposium on Foundations of Computer Science, IEEE*, pages 235–244, 2010.

[61] I. Koutis, G.L. Miller, and R. Peng. A nearly-$m \log n$ time solver for sdd linear systems. *IEEE 52nt Annual Symposium on Fouundations of Computer Science, IEEE*, pages 590–598, 2011.

[62] I. Koutis, G.L. Miller, and R. Peng. A fast solver for a class of linear systems. *Comunications of the ACM*, 55(10):99–107, 2012.

[63] J. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7:48–50, 1956.

[64] A. Leal-Duarte and C.R. Johnson. On the minimum number of distinct eigenvalues for a symmetric matrix whose graph is a given tree. *Mathematical Inequalities and Applications*, 5:175–180, 2002.

[65] Y.T. Lee and A. Sidford. Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems. *54th Annual Symposium on Foundations of Computer Science (FOCS), IEEE*, pages 147–156, 2013.

[66] C.-T. Lin. Structural controllability. *IEEE Transactions on Automatic Control*, AC-19(3):201–208, 1974.

[67] Y.-Y. Liu, J.-J. Slotine, and A.-L. Barabási. Controllability of complex networks. *Nature*, 473(7346):167–173, 2011.

[68] S.L. Ma and W.C. Waterhouse. The g-circulant solutions of $a^m = \lambda j$. *Linear Algebra and its Applications*, 85:211–220, 1987.

[69] M. Malyshev and V.E. Tarakanov. Generalized De Bruijn graphs. *Mathematical Notes*, 62(4):449–456, 1997.

[70] S. Martini, M. Egerstedt, and A. Bicchi. Controllability analysis of multi-agent systems using relaxed equitable partitions. *International Journal of Systems, Control and Communications*, 2(11213):100–121, 2010.

[71] H. Mayeda and T. Yamada. Strong structural controllability. *SIAM Journal on Control and Optimization*, 17(1):123–138, 1979.

[72] B.D. McKay. On the spectral characterization of trees. *Ars Combinatorica*, 3:219–232, 1979.

[73] N.S. Mendelsohn. An application of matrix theory to a problem in universal algebra. *Linear Algebra and its Applications*, 1:471–478, 1968.

[74] R. Merris. Laplacian matrices of graphs: A survey. *Linear Algebra and its Applications*, 197, 198:143–176, 1994.

[75] M. Mesbahi and M. Egerstedt. *Graph theoretic methods in multi-agent networks*. Princeton University Press, Princeton, 2010.

[76] S. Micali and V.V. Vazirani. An $\mathcal{O}(\sqrt{|V|}|e|)$ algorithm for finding maximum matching in general graphs. *Proc. 21st IEEE Symp. Foundations of Computer Science*, (17-27), 1980.

[77] N. Monshizadeh, S. Zhang, and M.K. Camlibel. Zero forcing sets and controllability of dynamical systems defined on graphs. *IEEE Trans. on Automatic Control*, 59(9):2562–2567, 2014.

[78] M. Nabi-Abdolyousefi and M. Mesbahi. On the controllability properties of circulant networks. *IEEE Trans. on Automatic Control*, 58(12):3179–3184, 2013.

[79] P.M. Nylen. Minimum-rank matrices with prescribed graph. *Linear Algebra and its Applications*, 248:303–316, 1996.

[80] D.D. Olesky, M. Tsatsomeros, and van den Driessche. Qualitative controllability and uncontrollability by a single entry. *Linear Algebra and its Applications*, 187:183–194, 1993.

[81] G. Parlangeli and G. Notarstefano. On the reachability and observability of path and cycle graphs. *IEEE Trans. on Automatic Control*, 57(3):743–748, 2012.

[82] J. Plesník. The NP-completeness of the hamiltonian cycle problem in planar digraphs with degree bound two. *Information Processing Letters*, 8(4):199–201, 1979.

[83] A. Rahmani, M. Ji, M. Mesbahi, and M. Egerstedt. Controllability of multi-agent systems from a graph theoretic perspective. *SIAM Journal on Control and Optimization*, 48(1):162–186, 2009.

[84] J.R. Riehl and M. Cao. Minimal-agent control of evolutionary games on tree networks. *The* 21^{st} *International Symposium on Mathematical Theory of Networks and Systems*, 2014.

[85] Y. Saad. *Iterative methods for sparse linear systems*. Siam, 2003.

[86] F.C. Santos, M.D. Santos, and J.M. Pacheco. Social diversity promotes the emergence of cooperation in public good games. *Nature*, 454(7201):213–216, 2008.

[87] C. Savage. Maximum matchings and trees. *Information Processing Letters*, 10(4-5):202–205, 1980.

[88] A. Schrijver. On the history of the transportation and maximum flow problems. *Mathematical Programming*, 91(3):437–445, 2002.

[89] A.J. Schwenk. *Almost all trees are cospectral, in New Directions in the Theory of Graphs*, pages 275–307. Academic Press, New York, 1973.

[90] D.A. Spielman and S.-H. Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. *Proceedings of the thirty-sixth annual ACM symposium on Theory of Computing, ACM*, pages 81–90, 2004.

[91] W. Stein. *Sage: Open Source Mathematical Software (Version 3.0.1)*, the sage group, 2008, http://www.sagemath.org.

[92] J. J. Sylvester. Chemistry and algebra. *Nature*, 17:284, 1878.

[93] H.G. Tanner. On the controllability of nearest neighbor interconnections. *Proc. of the 43rd IEEE conference on Decision and Control*, pages 2467–2472, 2004.

[94] M. Trefois and J.-C. Delvenne. Zero forcing number, constrained matchings and strong structural controllability. *Linear Algebra and its Applications*, 484:199–218, 2015.

[95] M. Trefois, M.T. Schaub, J.-C. Delvenne, and P. Van Dooren. A new sparse matrix factorization for linear solvers with fast iterations. *Submitted*, 2016.

[96] M. Trefois, P. Van Dooren, and J.-C. Delvenne. Binary factorizations of the matrix of all ones. *Linear Algebra and its Applications*, 468:63–79, 2015.

[97] E.R. van Dam and W.H. Haemers. Which graphs are determined by their spectrum ? *Linear Algebra and its Applications*, 373:241–272, 2003.

[98] R. Vandebril, M. Van Barel, and N. Mastronardi. *Matrix Computations and Semiseparable Matrices*, volume 1. Johns Hopkins University Press, 2007.

[99] R. Vandebril, M. Van Barel, and N. Mastronardi. *Matrix Computations and Semiseparable Matrices*, volume 2. Johns Hopkins University Press, 2008.

[100] K. Wang. On the g-circulant solutions to the matrix equation $A^m = \lambda J$. *Journal of Combinatorial Theory Ser. A*, 33:287–296, 1982.

[101] Y. Wu, R. Jia, and Q. Li. g-circulant solutions to the (0,1) matrix equation $A^m = J_n$. *Linear Algebra and its Applications*, 345:195–224, 2002.

[102] F. Yang. *Construction and application of hierarchical matrix preconditioners*. PhD thesis, University of Iowa, 2008.

[103] A.Y. Yazicioglu, W. Abbas, and M. Egerstedt. A tight lower bound on the controllability of networks with multiple leaders. *Proc. of the 51st IEEE conference on Decision and Control*, pages 1978–1983, 2012.

[104] D.M. Young. *Iterative solution of large linear systems*. Elsevier, 2014.

[105] S. Zhang, M.K. Camlibel, and M. Cao. Controllability of diffusively-coupled multi-agent systems with general and distance regular topologies. *Proc. of the 50th IEEE conference on Decision and Control and 2011 European Control Conference*, pages 759–764, 2011.